Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Miroslav Madon

Razvoj aplikacije za načrtovanje nadzemnih elektroenergetskih vodov - Electra

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: doc. dr. Luka Šajn

Ljubljana 2015

Rezultati diplomskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Študent naj v diplomskem delu predstavi programske rešitve pri načrtovanju nadzemnih elektroenergetskih vodov. Preuči naj obstoječe rešitve, ki to problematiko naslavljajo. Na tipičnem primeru naj predstavi tudi uporabo programa Electra. Opiše naj aplikacijo Electra in dodatke, ki jih je sam razvil. Predstavi naj tehnologije in programska orodja, ki se uporabljajo pri razvoju aplikacije, ter platforme, na katerih se program izvaja. Predstavi naj tudi implementacijo uporabniškega vmesnika s poudarkom na interaktivni risbi. Poda naj tudi teoretične osnove za izračun povesne verižnice. Predstavi naj tudi tipičen potek uporabe programa, kjer je skozi uporabniški vmesnik predstavljena uporaba implementirane funkcionalnosti.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Miroslav Madon, z vpisno številko **63960087**, sem avtor diplomskega dela z naslovom:

Razvoj aplikacije za načrtovanje nadzemnih elektroenergetskih vodov - Electra

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Luka Šajna,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 18. marca 2015 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Luku Šajnu za prijaznost, popravke in koristne nasvete pri izdelavi diplomske naloge. Zahvaljujem se tudi podjetju CGS plus, da mi je omogočilo izdelavo naloge v okviru razvojnega projekta podjetja. Posebna zahvala gre mojim staršem, ki so mi omogočili študij, ter moji Kseniji in sinu Aleksandru za podporo in potrpežljivost ob pisanju diplome.

Kazalo

Povzetek

V diplomskem delu so predstavljene programske rešitve pri razvoju programa Electra za načrtovanje nadzemnih elektroenergetskih vodov. Na tipičnem primeru je predstavljena tudi uporaba programa Electra.
V uvodu so podani motivi za razvoj programa Electra ter problematika, ki jo program naslavlja in rešuje.
V drugem poglavju so predstavljene tehnologije in programska orodja, ki smo jih uporabili pri razvoju aplikacije, ter platforme, na katerih se program Electra izvaja.
V tretjem poglavju je nekoliko podrobneje predstavljeno področje, ki se ukvarja z načrtovanjem nadzemnih elektroenergetskih vodov. Na kratko so opisane nekatere obstoječe rešitve, ki to problematiko naslavljajo. Predstavljen je tudi osnovni namen aplikacije Electra ter zahtevana funkcionalnost.
V četrtem poglavju sta podrobneje predstavljena razvoj ter delovanje aplikacije. Predstavljena je implementacija uporabniškega vmesnika s poudarkom na interaktivni risbi. Opisani so implementacija podatkovnega sloja aplikacije ter razredi za realizacijo numeričnih izračunov. Podane so teoretične osnove za izračun povesne verižnice. V sklepnem delu četrtega poglavja je opisan še tipičen potek uporabe programa, kjer je skozi uporabniški vmesnik predstavljena uporaba implementirane funkcionalnosti.
V zaključnem poglavju so predstavljene sklepne ugotovotve ter nekatere možnosti izboljšav.
Ključne besede: načrtovanje, projektiranje, nadzemni elektroenergetski vodi, Autocad, C++, ObjectArx, Electra

Abstract

In the thesis we present some technical solutions concerning the development of program Electra, which is used for overhead power lines planning and design. We also describe a typical example for the usage of Electra program.
We start with the motives for the development of the program and the issues Electra addresses and solves .
In the second chapter the technology and program tools that were used for the development of application and the platforms on which the Electra program runs are given.
Chapter No.3 presents the field that deals with overhead power lines planning and construction. Some actual software solutions available on the market that adress the problem are briefed as well. The basic purpose of Electra application is presented as well as the demanded functionality of it.
In the fourth chapter the development and functioning of application are detailed. The implementation of user interfaces with the stress on the interactive drawing is shown here alongside with the implementation of the data layer of application and the classes used for numerical calculations. Theoretical basis for catenary calculation is given. The second half of chapter No.4 contains a description of typical process for program usage. The use of implemented functionality is presented through the user interface.
We conclude with the final establisments and some possible improvements.

Keywords: planning, design, overhead power lines, Autocad, C++, ObjectArx, Electra

Seznam uporabljenih kratic in simbolov

- MFC (angl. *Microsoft Foundation Classes*) - Microsoftovi temeljni razredi sestavljajo knjižnico C++ razredov, ki ponuja objektno orientirano ovojnico za večino vmesnika Windows API.
- ARX (angl. *Autocad Runtime eXtension*) - razširitve v času izvajanja programa Autocad. Programski vmesnik do teh razširitev je ObjectArx.
- BRX (angl. *Bricscad Runtime eXtension*) - razširitve v času izvajanja programa Bricscad. Programski vmesnik do teh razširitev je BRX API.
- CAD (angl. *Computer Aided Design*) - računalniško podprto načrtovanje
- API (angl. *Application Programming Interface*) - programski vmesnik
- XML (angl. *Extensible Markup Language*) - razširljiv označevalni jezik
- DWG (angl. *DraWinG*) - binarna datoteka za shranjevanje podatkov in metapodatkov o 2D in 3D načrtih.Služi kot osnovna podatkovna baza za številne CAD platforme, med drugim tudi Autocad in Bricscad.
- CUI (angl. *Custom User Interface*) - sistem za prilagajanje uporabniških menijev in trakov v CAD programih po meri uporabnika ali uporabniških programov.

Poglavje 1

Uvod

Elektroenergetsko omrežje služi kot prenašalec električne energije od mesta proizvodnje do mesta porabe. Ker je v večini primerov mesto porabe precej oddaljeno od mesta proizvodnje, se je pojavila potreba po električnih daljnovodih, ki so v našem prostoru najpogosteje realizirani kot nadzemni elektroenergetski vodi. Nadzemni elektroenergetski vod je gradbeni objekt, ki je v grobem sestavljen iz vodnikov, izolatorskih sklopov in podpornih drogov s temelji. Nas bodo v pričujočem diplomskem delu bolj kot električne značilnosti daljnovoda zanimale mehanske značilnosti ter umestitev v prostor. Mehanske značilnosti so določene predvsem s podpornimi drogovi, obremenjenimi z vodniki in naravnimi dejavniki okolja, umestitev v prostor pa je pogojena s konfiguracijo terena, poseljenostjo ter namembnostjo zemljišč, po katerih poteka, kakor tudi s sprejemanjem kompromisov med raznimi družbenemi in ekonomskimi interesi in seveda s čim nižjimi stroški izgradnje, obratovanja in vzdrževanja. Ob vsem naštetem je potrebno upoštevati še stroge standarde, ki z določili zajemajo področja od varstva okolja in narave, prostorskega načrtovanja in elektromagnetnega sevanja, pa do čisto gradbenih določil, ki morajo upoštevati obremenitve drogov in vodnikov v odvisnosti od pričakovanih podnebnih pogojev ter od uporabljenih materialov. Vse z namenom zagotavljanja zanesljivega in trajnega obratovanja nadzemnega elektroenergetskega voda ob upoštevanju vse predpisane zakonodaje. Načrtovanje

nadzemnih elektroenergetskih vodov zato že dolgo ni več trivialen proces in nič nenavadnega torej ni, da se je ob razvoju, predvsem računalniške tehnologije, pojavila potreba po uporabi programskih orodij, ki bi omogočala lažjo, hitrejšo, fleksibilnejšo in kar najbolj avtomatizirano izdelavo projekta ter predpisane projektne dokumentacije za izgradnjo nadzemnih elektroenergetskih vodov. V slovenskem prostoru vse do druge polovice devetdesetih let prejšnjega stoletja ni bilo računalniške programske opreme, ki bi reševala to problematiko. Podjetje "CGS plus" je konec devetdesetih let v sklopu svojih programskih rešitev za načrtovnje infrastrukturnih objektov izdalo modul za izris vzdolžnih profilov daljnovodov v programu Autocad. Od takrat pa vse do sedaj je bil ta modul za marsikaterega projektanta nadzemnih elektroenergetskih vodov nadvse dobrodošel in koristen pripomoček, ker pa so dandanes potrebe postale večje in drugačne, računalniška tehnologija pa omogoča več v krajšem času, se je tudi v podjetju "CGS plus" pojavila potreba po izdelavi sodobnejšega programa. Kot razvijalcu v podjetju "CGS plus" mi je bil zaupan razvoj novega programa in glavne funkcionalnosti, ki sem jih razvil, sem se odločil opisati v tem diplomskem delu.

Moj cilj je bil program, ki bo omogočal dinamičen in medsebojno povezan vnos in urejanje tlorisne trase daljnovoda, kot tudi vzdolžnega poteka daljnovoda ter sprotne izračune in izrise projektiranega stanja tako v tlorisu trase, kot tudi v vzdolžnem poteku ter pripadajočih tabelah. Podprto mora biti vnašanje in urejanje podpornih drogov in vodnikov tako v tlorisu trase kot v vzdolžnem poteku, pri tem pa se morajo poteki vodnikov v obliki povesne verižnice in natezne sile, ki delujejo na drogove, računati sproti ob vsaki spremembi daljnovoda. Potek terena v prikazu vzdolžnega poteka mora v vsakem trenutku ustrezati projekciji trase na digitalni relief modela, po katerem trasa poteka.

Zastavljen cilj sem v okviru tega diplomskega dela dosegel tako, da sem razvil program, ki deluje na platformah Autocad in Bricscad ter uporablja risbo DWG za izris, vnos, urejanje in hranjenje podatkov o daljnovodu. Za dinamičnost in sprotno računanje ter prikaz sem uporabil funkcionalnosti Au-

tocada in Bricscada, ki omogočata proženje reaktorjev in sprotno računanje in predogled izrisa glede na trenutni položaj kazalca miške. Program sem zasnoval kot modul programske opreme za nizkogradnjo "CGS Infrastructure Design Suite" podjetja "CGS plus" zato sem lahko uporabil tudi marsikatero funkcionalnost iz obstoječih modulov za načrtovnje infrastrukturnih objektov.

Poglavje 2

Uporabljene tehnologije in orodja

Aplikacija za načrtovanje nadzemnih elektroenergetskih vodov Electra je bila v celoti izdelana v programskem jeziku C++ z uporabo razvojnega okolja Visual Studio 2010. Za izdelavo pogovornih oken ter za delo s tekstovnimi nizi in podatkovnimi tabelami je bila uporabljena Microsoftova knjižnica MFC. Program je zasnovan za delovanje na platformah CAD, od katerih sta podprti Autocad in Bricscad. Programski modul se v obliki dinamične knjižnice naloži v ustrezno CAD platformo, na kateri se potem izvaja in deluje kot njena ekstenzija. V ta namen je potrebno pri razvoju uporabiti ustrezno knjižnico kot programski vmesnik za interakcijo z izbrano platformo, in sicer Object ARX za platformo Autocad ter BRX API za Bricscad. Program Electra je zasnovan kot modul v okviru programskega paketa za nizkogradnjo "CGS Infrastructure Design Suite" podjetja "CGS plus", zato za svoje delovanje uporablja tudi nekatere funkcionalnosti, ki se nahajajo v skupnem jedru vseh modulov.

2.1 C++

C++ je splošno namenski programski jezik, ki se prevede v strojni jezik, kar pomeni veliko hitrejše izvajanje aplikacij v primerjavi s tistimi, ki so napisane v jeziku, ki se interpretira. Omogoča proceduralen ali objektno usmerjen programski pristop, zato je izredno fleksibilen za uporabo in kot tak še vedno eden najbolj priljubljenih in uporabljanih programskih jezikov. Tudi programa Autocad in Bricscad ter njuni programski knjižnici ObjectARX in BRX API so napisani v programskem jeziku C++, tako da je za kvaliteten razvoj programske opreme za te platforme uporaba programskega jezika C++ zelo priporočljiva.

2.2 Microsoft Visual Studio 2010

Microsoft Visual Studio je integrirano razvojno okolje, ki se uporablja za razvoj konzolnih aplikacij in aplikacij z grafičnim uporabniškim vmesnikom. Vključuje urejevalnik izvorne kode, urejevalnik grafičnega uporabniškega vmesnika, integrirani razhroščevalnik ter prevajalnik. Poleg tega vključuje še celo vrsto dodatnih razvojnih orodij, ki proces razvoja, razhroščevanja in testiranja še dodatno olajšajo ter pospešijo. Namenjen je razvoju aplikacij, ki se izvajajo v okoljih Microsoft Windows, Windows Mobile, Windows CE, .NET Framework ter Microsoft Silverlight. Podpira razvoj v različnih programskih jezikih. Za namen te diplomske naloge sem ga uporabljal kot razvojno okolje za programski jezik C++.

2.3 MFC

Knjižnica MFC ponuja objektno orientirano ovojnico za večino programskega vmesnika Windows API v obliki C++ razredov. Razredi so definirani za številne Windows objekte, okna ter navadne gradnike. Knjižnica MFC je zelo uporabna tudi za delo s tekstovnimi nizi ter podatkovnimi tabelami in seznami. Kljub nekaterim pomanjkljivostim je njena uporaba zelo priporočljiva

za razvoj programov na platformah Autocad in Bricscad, saj imata oba uporabniški vmesnik narejen s knjižnico MFC. Poleg tega so številni razredi v njunih knjižnicah Object ARX in BRX API izpeljani neposredno iz MFC razredov.

2.4 Autocad

Autocad je računalniška aplikacija podjetja Autodesk, ki je namenjena 2D in 3D računalniško podprtemu načrtovanju. Deluje na operacijskih sistemih Microsoft Windows in Mac OS, od leta 2010 pa obstaja tudi kot mobilna aplikacija za Android. Autocad podpira številne programske vmesnike za prirejanje in avtomatizacijo, med katerimi je najbolj uporabljan in razširjen ObjectARX. Zaradi tega dejstva Autocad že dolgo ni več samo samostojni program, ampak je tudi platforma za številne aplikacije.

2.5 Bricscad

Bricscad je računalniška aplikacija podjetja Bricsys, ki je namenjena 2D in 3D računalniško podprtemu načrtovanju. Je v večini najpogostejših funkcionalnosti povsem enakovredna Autocad-u. Enako velja za njen programski vmesnik BRX API. Zaradi precej nižje cene je vedno pogostejša alternativa Autocad-u.

2.6 ObjectARX

ObjectARX je programski vmesnik za prirejanje, avtomatizacijo in razširitve programa Autocad. Sestavljajo ga glave z deklaracijami ter knjižnice, ki se uporabijo za generiranje datotek Windows DLL s končnico arx, ki se lahko naložijo v proces Autocad-a in omogočajo vzajemno delovanje z aplikacijo Autocad. Ravno tako omogoča direkten dostop do podatkovnih struktur Autocad-a in njegovega grafičnega sistema.

2.7 BRX API

BRX API je programski vmesnik, ki je popolnoma enakovreden programskemu vmesniku ObjectARX, le da je narejen za Bricscad. (Za podrobnosti glej ObjectARX)

2.8 CUI

CUI je je Autocad-ova tehnologija, ki omogoča prilagajanje menijev in trakov potrebam uporabnika. To tehnologijo so privzeli tudi nekateri ostali programi CAD, med drugimi tudi Bricscad.

2.9 Programska oprema "CGS Infrastructure Design Suite"

"CGS Infrastructure Design Suite" je programska oprema podjetja CGS plus, ki je namenjena načrtovanju infrastrukturnih objektov. V programski paket so vključeni sledeči programski moduli:

- **PLATEIA** za načrtovanje cest
- **FERROVIA** za načrtovanje železnic
- **AQUATERRA** za načrtovanje in urejanje regulacij vodotokov
- **AUTOPATH** za načrtovanje zavijalnih krivulj in analize prevoznosti
- **ELECTRA** za načrtovanje nadzemnih elektroenergetskih vodov

Programska oprema "CGS Infrastructure Design Suite" deluje na platformah Autocad in Bricscad.

Poglavje 3

Opis problema

3.1 Problematika načrtovanja nadzemnih elektroenergetskih vodov

Da bi razvili kvaliteten program za načrtovanje nadzemnih elektroenergetskih vodov, moramo seveda poznati problematiko, s katero se srečujejo projektanti. Projektant dobi nalogo speljati daljnovod iz zahtevane začetne točke v zahtevano končno točko. Pri tem ima vnaprej določen koridor, po katerem mora potekati trasa daljnovoda. Iz geodetskih služb dobi posnetke obstoječega terena, po katerem naj bi potekala trasa daljnovoda. Geodetski posnetki so dandanes v elektronski obliki, predstavljeni kot množica višinskih točk, iz katerih se lahko kasneje z različnimi namenskimi računalniškimi orodji generira digitalni model terena. Generirani digitalni model terena se najpogosteje uporabi v različnih CAD orodjih, kjer je prikazan kot množica trikotnikov, ki tvorijo površino reliefa, ali pa kot množica plastnic, ki povezujejo točke reliefa z enako nadmorsko višino. Ta relief potem služi kot osnova za določanje višinskega poteka trase daljnovoda, ki se dobi tako, da se začrtana trasa projecira na površino. Ker bomo v nadaljnem besedilu večkrat uporabili pojem stacionaža, velja na tem mestu zapisati njegov pomen. Stacionaža je merska lestvica za označevanje dolžinskih položajev na trasi z opredeljenim začetkom, začetno vrednostjo in smerjo. Ko ima pro-

jektant definiran tlorisni potek trase ter višinski potek trase, lahko začne na traso postavljati podporne drogove ter nanje napenjati ustrezne vrvi. V grobem ločimo tri vrste drogov, in sicer zatezne, kotne in nosilne. Vodnike oziroma vrvi se napenja med dva zatezna ali kotna droga, s čimer se ustvari zatezno polje. Vmes se za podporo vodnikov postavlja nosilne drogove, ki pa nimajo zatezne funkcije, ampak služijo le za nošenje vodnika. Na postavljene drogove je nato potrebno skozi definirane točke obesišča napeljati še zahtevane električne vodnike, ki nastopajo v obliki vrvi različnih tipov in sestav. Upoštevati je potrebno, da dobi vsaka vrv, ki je napeta med dvema zateznima drogovoma in podprta z nosilnimi drogovi, obliko povesne verižnice, katere oblika je odvisna od specifične teže vrvi, faktorja elastičnosti, preseka vrvi ter dodatnega zimskega bremena. Povesno verižnico je potrebno izrisati za primer največjega povesa, saj je pri načrtovanju daljnovoda potrebno upoštevati predpisane minimalne varnostne razdalje med vodniki in terenom oziroma obstoječimi objekti. Ko projektant z upoštevanjem vseh omenjenih predpisov in omejitev začrta trasni potek ter vzdolžni potek nadzemnega elektroenergetskega voda, s tem delo še ni zaključeno. Za projektirano traso mora izdelati še tabelo vzdolžnega profila z vsemi kotami in stacionažami daljnovoda, trasni načrt daljnovoda ter nazadnje še montažne sheme, kjer so za vsako zatezno polje in za vsak vodnik natančno izračunane vse natezne napetosti ter povesi vodnikov pri vseh temperaturah od $-20\,^{\circ}\mathrm{C}$ do $+40\,^{\circ}\mathrm{C}$ stopinj celzija, pri $-5\,^{\circ}\mathrm{C}$ pa še za primer dodatnega zimskega bremena na vodnikih. Treba je poudariti, da je pri vsaki spremembi na trasi daljnovoda, položajev drogov, tipov vrvi ali kateregakoli drugega parametra, potrebno ponovno izrisati vse situacijske in vzdolžne poteke ter ponovno izpolniti vse pripadajoče tabele ter opraviti nove izračune vseh podatkov v montažnih shemah. Opisana problematika je zelo zgovoren motiv za izdelavo programa, ki bi lahko kar največji del opisanega procesa avtomatiziral, pospešil in omogočil čimbolj intuitivno in pregledno izdelavo in vzdrževanje načrta nadzemnega elektroenergetskega voda.

3.2 Obstoječe rešitve

V Sloveniji si projektanti pri načrtovanju nadzemnih elektroenergetskih vodov pomagajo z različnimi orodji. Za manjše nizkonapetostne in srednjenapetostne daljnovode velikokrat uporabljajo ločene programske pripomočke za računanje povesov vodnikov, nateznih sil in ostalih podatkov, potrebnih za izdelavo montažnih shem in vzdolžnih potekov. Na osnovi teh podatkov morajo nato v enem izmed programov CAD narisati vzdolžne poteke, ki jih zahteva projektna dokumentacija. V tem smislu ne moremo govoriti o pravem programskem orodju za načrtovanje, saj takšni pripomočki ne delujejo med seboj povezano nad enotnimi in trajnimi podatki daljnovoda z možnostjo vizualizacije, ampak ponavadi služijo le za opravljanje izračunov posameznih parametrov. Seveda pa se za načrtovanje večjih in kompleksnejših daljnovodov, največkrat visokonapetostnih, uporablja ena izmed profesionalnih programskih oprem. Na svetovnem trgu je kar nekaj programskih rešitev, ki se spopadajo s problematiko načrtovanja nadzemnih elektroenergetskih vodov, ki pa se zelo razlikujejo po kompleksnosti in obsežnosti. Na tem mestu jih bom naštel le nekaj.

PLS-CADD je eno izmed najprofesionalnejših in najbolj razširjenih programskih orodij na svetovnem trgu, je pa tudi eno izmed najdražjih, zato se ponavadi uporablja le za projektiranje največjih in najkompleksnejših nadzemnih elektroenergetskih omrežij. Omogoča integracijo z GIS sistemi, napredno 3D vizualizacijo ter možnost izvoza podatkov v programska orodja CAD. Programska oprema je izdelek ameriškega podjetja Powerline systems. [5]

CATAN je produkt avstralskega podjetja Alliance Power and Data (APD). Gre za relativno preprost program, ki je namenjen načrtovanju manj kompleksnih daljnovodov. Omogoča večino najosnovnejših izračunov, ki se izvajajo med samim urejanjem. Za vizualizacijo ima vgrajen svoj grafični vmesnik ter omogoča izvoz podatkov v DXF. [6] Iz Avstralije prihajata tudi programa LIVEWIRE ter Poles 'n' Wires, ki sta po kompleksnosti in funkcionalnosti primerljiva s programom CATAN. Oba omogočata izvoz vzdolžnega poteka v DXF za kasnejšo vključitev v enega izmed programov CAD. [7] [8] Slo-

venski predstavnik programa za načrtovanje nadzemnih elektroenergetskih vodov je modul za načrtovanje vzdolžnih potekov nadzemnih elektroenergetskih vodov podjetja CGS plus, ki je predhodnik programa, ki smo ga razvili v okviru pričujoče diplomske naloge. Program omogoča izris vzdolžnega poteka direktno v risbi CAD. Omogoča izračun povesne verižnice, vendar le za en vodnik za vsak vzdolžni potek. Omogoča tudi generiranje montažnih tabel v tekstovni obliki. Omejitev programa je, da ne omogoča dinamičnega spreminjanja vzdolžnega poteka in trase daljnovoda, ter to, da se parametri daljnovoda ne izračunavajo sproti med samim vnašanjem drogov, ampak je potrebno izračune poganjati posebej. Te omejitve so bile tudi glavni vzrok za zasnovo novega programa, v katerem smo te pomanjkljivosti odpravili, hkrati pa razvili nove funkcionalnosti in nadgradili obstoječe.

3.3 Namen aplikacije in zahtevana funkcionalnost

Novo aplikacijo za načrtovanje nadzemnih elektroenergetskih vodov "Electra" smo razvili z namenom, da bi projektantom omogočili čimbolj avtomatizirano in interaktivno izdelavo in vzdrževanje načrta trase ter vzdolžnega poteka daljnovoda. Aplikacija je namenjena izdelavi gradbenega dela projektnega načrta za daljnovod in za samo umestitev daljnovoda v prostor.

Zahtevana funkcionalnost aplikacije:

- delovanje na platformah Autocad in Bricscad
- možnost vnosa trase na digitalni relief terena in avtomatsko projiciranje trase na površino
- možnost avtomatskga izrisa vzdolžnega poteka daljnovoda, na podlagi vnešene trase

- avtomatska postavitev začetnega in končnega zateznega droga ter vseh kotnih drogov na vsako točko, kjer trasa spremeni smer
- možnost vnosa dodatnih zateznih ali nosilnih drogov, tako v situativnem, kot v vzdolžnem pogledu
- možnost definiranja poljubnega števila vodnikov in njihovih obesišč na vsakem zateznem polju
- avtomatski izračun in izris poteka vodnikov v vzdolžnem pogledu v obliki povesne verižnice ter prikaz varnostne višine
- Vsak premik polilinije, ki predstavlja potek trase v situativnem pogledu, se mora avtomatsko odražati v ustrezno spremenjenem vzdolžnem poteku daljnovoda.
- možnost interaktivnega določanja stacionaže drogov z vlečenjem miške in sproten prikaz povesov vseh definiranih vrvi
- avtomatska izdelava in ažuriranje podatkov v tabeli vzdolžnega poteka
- možnost izračuna in izdelave predpisanih montažnih tabel v obliki tekstovnih datotek

Poglavje 4

Razvoj in delovanje aplikacije

4.1 Uporabniški vmesnik

Uporabniški vmesnik programa Electra sestavljajo meniji in orodni trak, pogovorna okna, ukazna vrstica ter interaktivna risba. Ravno z implementacijo interaktivne risbe smo nadgradili klasične elemente uporabniškega vmesnika in dosegli željeno dinamičnost in preglednost pri delu s programom, zato bomo v nadaljevanju razvoju tega elementa uporabniškega vmesnika namenili največ pozornosti.

4.1.1 Meniji in orodni trak

Prvi od načinov interakcije s programom je uporaba menijev in orodnega traku. Meni ter orodni trak programa Electra sta vgrajena v sklop menijev CGS Infrastructure Design Suite. Autocad platforma omogoča kreiranje ter umeščanje menijev in orodnega traku med ostale Autocadove menije in orodne trakove. To je omogočeno preko Autocadovega sistema za prilagajanje uporabniškega vmesnika CUI. Do orodja za urejanje menijev in orodnih trakov dostopamo tako, da prek ukazne vrstice programa Autocad poženemo ukaz CUI. Pojavi se uporabniški vmesnik, v katerem lahko ustvarjamo nove menije in orodne trakove ali pa prilagajamo obstoječe. Omogočeno nam je določanje celotnega izgleda menijev in orodnih trakov ter določanje ukazov,

ki se poženejo ob posamezni menijski izbiri. Imena funkcij, ki jih želimo poganjati iz menijev in orodnih trakov, moramo v programski kodi ostrezno registrirati in izvoziti. V programskem vmesniku ObjectARX imamo v ta namen pripravljen C++ makro. Na primeru 4.1 si lahko ogledamo, kako smo izvozili funkcijo za urejanje drogov EL_EDITPOST.

```
ACED_ARXCOMMAND_ENTRY_AUTO(CElectraNewApp, CGSElectraNew, _EL_EDITPOST, EL_EDITPOST,
    ACRX_CMD_TRANSPARENT, NULL)
```

Primer 4.1: Registracija funkcije za dostopanje iz Autocada

Ustvarjeni skupek menijev in orodnih trakov shranimo v datoteko tipa cui, ki jo nato distribuiramo skupaj s programom in ob zagonu našega programa v Autocadu z ustrezno lispovo proceduro tudi naložimo. Datoteka cui je v formatu xml, ki ga zna Autocad interpertirati in prikazati kot meni ali orodni trak. V podrobnosti procesa ustvarjanja in kreiranja menijev ter zgradbe datotek cui se v diplomski nalogi ne bomo spuščali.

4.1.2 Pogovorna okna

V programu Electra so pogovorna okna uporabljena za opravila kot so definiranje drogov, definiranje vrvi, definiranje parametrov verižnic, shranjevanje poročil in ostala opravila, kjer smo ocenili, da je pogovorno okno najprimernejša interakcija s programom. Za izdelavo pogovornih oken smo uporabili funkcionalnosti za izdelavo uporabniškega vmesnika, ki ga ponuja programski vmesnik ObjectARX. ObjectARX namreč ponuja množico razredov AcUi izpeljanih iz MFC razredov, s katerimi imamo možnost implementirati uporabniški vmesnik, ki je funkcionalno in vizualno konsistenten z Autocadovim uporabniškim vmesnikom.[4] Razredi AcUi so kompatibilni z MFC razredi, zato smo, kjer se nam je zdelo primerneje, uporabili kombinacijo obeh. V programu Electra smo za implementacijo pogovornih oken uporabili sledeče razrede:

- *CAcUiDialog* za implementacijo pogovornega okna
- *CStatic* za implementacijo statičnih tekstovnih nizov v pogovornem oknu

- *CButton* za implementacijo navadnih gumbov v pogovornem oknu
- *CAcUiPickButton* za implementacijo gumbov, ki vsebujejo sliko.
- *CAcUiNumericEdit* za implementacijo vnosnih polj, ki omogočajo avtomatsko preverjanje številčnih vrednosti
- *CAcUiEdit* za implementacijo splošnih vnosnih polj v pogovornem oknu
- *CComboBox* za implementacijo padajočih izbirnih menijev v pogovornem oknu
- *CGridCtrl* za implementacijo tabele znotraj pogovornega okna

4.1.3 Ukazna vrstica

Kljub temu da imamo na voljo več načinov za vizualno bogat uporabniški vmesnik, pa je za nekatera opravila še vedno najprimernejša in tudi najhitrejša uporaba ukazne vrstice. Uporabniki Autocada zelo velik del interakcije s programom opravijo preko ukazne vrstice, zato smo se je v nekaterih primerih poslužili tudi v programu Electra. Za implementacijo interakcije prek ukazne vrstice smo uporabili funkcije, ki jih nudi programski vmesnik ObjectARX [4]:

- *acutPrintf* za izpisovanje poljubnega teksta v ukazno vrstico
- *acedGetInt* za povpraševanje po celoštevilski vrednosti
- *acedGetReal* za povpraševanje po realni vrednosti
- *acedGetDist* za povpraševanje po razdalji
- *acedGetAngle* za povpraševanje po kotu
- *acedGetPoint* za povpraševanje po točki
- *acedGetKword* za povpraševanje po ključni besedi
- *acedGetString* za povpraševanje po tekstovnem nizu
- *acedEntSel()* za povpraševanje po entiteti

Funkcije za uporabnikov vnos vrednosti, ki imajo obliko acedGetXXX(), počakajo uporabnika, da vnese vrednost ustreznega tipa in vrnejo rezultat v rezultirajoči argument. Pri uporabi teh funkcij lahko definiramo tudi poljuben tekst, ki uporabnika povpraša po vrednosti preden počaka na vnos.

Podobno se obnaša tudi funkcija acedEntSel, ki povpraša in počaka uporabnika, da izbere ustrezno entiteto v risbi. Večina funkcij za uporabnikov vnos podpira več načinov obnašanja, zato imamo na voljo funkcijo acedInitGet(), ki jo z ustreznimi parametri pokličemo pred funkcijo za vnos ter tako določimo njen način delovanja.[4] Primer uporabe ukazne vrstice pri interakciji z uporabnikom je razviden na sliki 4.1, kjer je prikazan vnos točk vzdolž trase, v katerih naj se tabelira višina vrvi nad terenom.

Slika 4.1: Uporaba ukazne vrstice pri tabeliranju varnostne višine.

4.1.4 Interaktivna risba

Uporaba menijev, trakov, ukazne vrstice in pogovornih oken je najbolj klasičen način interakcije s programom, vendar pa si pri ustvarjanju in urejanju tehničnih risb velikokrat želimo neposredne interakcije z vizualiziranimi podatki na zaslonu, saj je tako vsaj v prvi fazi načrtovanja delo bolj intuitivno, manj zamudno in omogoča večjo ustvarjalnost. Zagotavljanje takšnega načina dela je bil eden od naših prednostnih ciljev pri izdelavi programa Electra, zato smo se odločili izkoristiti eno od poglavitnih prednosti uporabe platforme CAD za izvajanje programa, ki je uporaba same risbe DWG za uporabniški vmesnik, preko katerega so mogoči vnos, urejanje in vizualizacija podatkov. Za realizacijo tega cilja smo uporabili razpoložljive vmesnike in orodja, ki nam jih ponuja programski vmesnik ObjectARX. Ta vmesnik nam ponuja poleg ševilnih osnovnih funkcij namenjenih vnosu, izbiri ali urejanju entitet tudi nekatere naprednejše funkcionalnosti, ki nam omogočajo implementacijo bolj dinamičnih načinov vnosa in urejanja podatkov. Od slednjih smo mi uporabili Autocadove reaktorje in Jig.

Vnos entitete

Program smo zasnovali tako, da se pri vnosu ali izračunu podatkov o daljnovodu hkrati tudi vnesejo vizualne reprezentacije teh podatkov v risbo. Ko na primer vnesemo podatke o novem drogu daljnovoda, se na definirani stacionaži na osi v situaciji izriše krogec, v vzdolžnem profilu pa navpična linija z ustrezno višino in definiranimi opisi. Podobno se pri izračunu verižnice izriše polilinija, ki ponazarja potek povešenega vodnika in še bi lahko naštevali. Elemente, ki so vnešeni v podatkovno bazo risbe DWG in imajo lastnost grafičnega prikaza v risbi, imenujemo entitete. Za programsko kreiranje in vnos entitet v risbo smo uporabili razrede programskega vmesnika ObjectARX, ki so izpeljani iz razreda AcDbEntity. AcDbEntity je osnovni razred za vse objekte podatkovne baze DWG, ki imajo grafično upodobitev. [4] Nekaj tako izpeljanih razredov, ki smo jih mi najpogosteje uporabljali pri izdelavi programa, je:

- *AcDbLine* za grafični prikaz linije
- *AcDb3DPolyline* za grafični prikaz polilinije v 3D prostoru
- *AcDb2DPolyline* za grafični prikaz polilinije v 2D prostoru
- *AcDbCircle* za grafični prikaz kroga
- *AcDbText* za grafični prikaz teksta
- *AcDbBlockReference* za grafični prikaz bloka, ki je sestavljen iz množice poljubnih entitet in atributov, ali pa predstavlja kar drugo risbo DWG, ki se prikazuje v trenutni risbi z definirano pozicijo in atributi.

Primer 4.2 prikazuje, kako z uporabo vmesnika ObjectARX programsko izrišemo linijo v risbo DWG. Vidimo lahko, da je potrebno alocirati novo instanco razreda AcDbLine z začetno in končno točko ter jo dodati v bazo DWG, natančneje v tabelo simbolov v zapis ACDB_MODEL_SPACE. Ko entiteto pripnemo v bazo, ji sistem dodeli unikatni ključ tipa AcDbObjectId, ki nam lahko v nadalje služi za dostop do te entitete in branje ter spreminjanje njenih lastnosti.

```
AcDbObjectId drawLine() {
  AcDbBlockTable *pBlockTable;
  AcDbLine *pLine(NULL);
  AcDbBlockTableRecord *pBlockTableRecord;
  AcDbObjectId lineId;
  AcGePoint3d startPt(3.0, 5.0, 0.0), endPt(12.0, 8.0, 0.0);

  pLine = new AcDbLine(startPt, endPt);
  acdbHostApplicationServices()->workingDatabase()->getSymbolTable(pBlockTable, AcDb::
    kForRead);
  pBlockTable->getAt(ACDB_MODEL_SPACE, pBlockTableRecord, AcDb::kForWrite);
  pBlockTable->close();
  pBlockTableRecord->appendAcDbEntity(lineId, pLine);
  pBlockTableRecord->close();
  pLine->close();

  return lineId;
}
```

Primer 4.2: Izris linije

Izbira entitete

Entitete, ki smo jih izrisali v risbo, nam ne predstavljajo samo izrisa elementov daljnovoda, temveč tudi grafično ponazoritev vnešenih podatkov o nekem elementu daljnovoda. Prvi korak k temu, da bi lahko posamezne objekte daljnovoda urejali interaktivno z miško v risbi DWG, je omogočiti uporabniku izbiranje entitet. To smo realizirali z uporabo funkcije acedEntSel, ki nam omogoča, da v ukazno vrstico izpišemo navodilo uporabniku, hkrati pa mu vklopimo kazalec za izbiro entitete. Rezultat te funkcije je ime izbrane entitete s pomočjo katerega lahko dobimo AcDbObjectId entitete, tega pa lahko uporabimo za odpiranje entitete in branje ali spreminjanje njenih lastnosti.

```
AcDbObjectId Functions::selectLine() {
  ads_name eName; ads_point pt;
  AcDbObject *pObj(NULL);
  AcDbObjectId oid; oid.setNull();

  if (acedEntSel(_T("\nIzberi linijo:"), eName, pt)==RTNORM) {
    if (acdbGetObjectId(oid, eName)==Acad::eOk) { // Preberemo id od izbrane entitete
      //izbrano entiteto odpremo za branje
      if ((acdbOpenObject(pObj, oid, AcDb::kForRead)==Acad::eOk) && (pObj!=NULL)) {
        if (pObj->isA()!=AcDbLine::desc()) { // Is the selected entity an AcDbPolyline
          oid.setNull(); //Ce entiteta ni AcDbLine, postavimo object id na NULL
        }
        pObj->close(); //Odprt objekt je potrebno vedno zapreti
      }
    }
  }  return oid;
}
```

Primer 4.3: Izbira linije

V primeru 4.3 lahko vidimo, kako smo z uporabo funkcije acedEntSel implementirali funkcijo za izbiro entitete tipa AcDbLine. Funkcija s sporočilom v ukazni vrstici nagovori uporabnika, naj izbere linijo, počaka na uporabnikovo izbiro in nato preveri, če je uporabnik izbral entiteto tipa AcDbLine. Če je temu tako, funkcija vrne ustrezen id. Na ekvivalenten način smo realizirali tudi izbire drugih tipov entitet.

Reaktorji

Naslednja tehnologija, ki smo jo izkoristili za interaktivno delo z risbo, so reaktorji. Reaktorji so neke vrste prekinitve, ki se prožijo ob določenih akcijah nad autocadovimi objekti ali nad samo risbo DWG in njeno bazo. Mi smo uporabili tip reaktorja, ki je vezan na entiteto in se proži ob vsakem njenem urejanju. Z njegovo uporabo smo dosegli, da se ob vsakem urejanju polilinije, ki predstavlja traso daljnovoda, avtomatsko pokličejo funkcije za ponoven preračun vseh podatkov daljnovoda ter posodobitev izrisa celotnega vzdolžnega poteka vključno z drogovi in vodniki. S tem smo uporabniku omogočili, da med enostavnim urejanjem autocadove polilinije sproti spremlja, kako se spremembe trase odražajo v vzdolžnem poteku daljnovoda. To je ena od pomembnih prednosti programa Electra, saj omogoča uporabniku, da v občutno krajšem času opravi več iteracij urejanja trase daljnovoda in tako prej pride do najustreznejše rešitve.

```
class ElectraAxesModifyReactor : public AcDbObject
{
  public:
    ACRX_DECLARE_MEMBERS(ElectraAxesModifyReactor);
    ElectraAxesModifyReactor();
    void eLinkage(AcDbObjectId i, int idx, double f=1.0) {mId=i; mFactor=f; m_axisIdx=
     idx;};
    void modified(const AcDbObject*);
    void openedForModify(const AcDbObject*);
    Acad::ErrorStatus dwgInFields(AcDbDwgFiler*);
    Acad::ErrorStatus dwgOutFields(AcDbDwgFiler*) const;
    Acad::ErrorStatus dxfInFields(AcDbDxfFiler*);
    Acad::ErrorStatus dxfOutFields(AcDbDxfFiler*) const;
  private:
    AcDbObjectId mId;
```

```
    double mFactor;
    int m_axisIdx;
    AcGePoint3dArray *m_prevPoints;
};

BOOL addReactorToEntity(AcDbObjectId oId, int idx);
```

Primer 4.4: definicija reaktorja, ki se proži ob spremembi trase

Na primeru 4.4 lahko vidimo, kako smo definirali reaktor, ki smo ga vezali na polilinijo, s katero je izrisana trasa daljnovoda. Definirali smo razred ElectraAxesModifyReactor, ki je izpeljan iz osnovnega razreda AcDbObject. Prekriti smo morali funkcije dwgInFields, dwgOutFields, dxfInFields in dxfOutFields, ki se kličejo ob shranjevanju in nalaganju objekta. To nam omogoča, da se naš reaktor shranjuje v bazo DWG in bere iz nje skupaj z entiteto, na katero smo ga vezali. Poleg teh funkcij pa imamo možnost prekrivanja še številnih notifikacijskih funkcij, ki se avtomatsko pokličejo ob določeni spremembi statusa entitete. Mi smo prekrili in uporabili notifikacijski funkciji openedForModify in modified. Prva se samodejno pokliče pred vsako spremembo entitete, druga pa po vsaki spremembi entitete. V primeru 4.5 je komentirano, kako smo funkciji uporabili v našem programu.

```
void ElectraAxesModifyReactor::openedForModify(const AcDbObject* pObj) {
  //pred vsako spremembo polilinije, ki predstavlja trenutni tlorisni potek trase,
  //si zapomnimo položaje vseh vozlišč polilinije, da bomo lahko ovrednotili spremembo
  //po končanem urejanju.
}
void ElectraAxesModifyReactor::modified(const AcDbObject* pObj) {
  //Po končanem urejanju naredimo:
  //-projiciranje spremenjene trase na digitalni relief terena
  //-pokličemo funkcijo, ki na podlagi primerjave spremenjene polilinije s shranjenimi
  //položaji točk pred urejanjem,
  //posodobi položaje obstoječih drogov, po potrebi vnese nove
  //ter ponovno preračuna potek vseh definiranih vrvi
  //-pokličemo funkcijo za posodobitev podatkov v razpredelnici vzdolžnega profila
}
```

Primer 4.5: Uporaba notifikacijskih funkcij openedForModify in modified

Ko imamo reaktor definiran, ga moramo samo še vezati na izbrano entiteto, v našem primeru na polilinijo, ki predstavlja tlorisni potek trase. V ta namen smo naredili funkcijo addReactorToEntity (glej Primer 4.6), ki vpiše reaktor v ustrezen slovar baze DWG za podano os ter veže reaktor na entiteto s podanim AcDbObjectId. To vezavo naredimo s funkcijo addPersistentReactor programskega vmesnika ObjectARX [4].

```
BOOL addReactorToEntity(AcDbObjectId oId, int idx)
{
  AcDbEntity *pEnt(NULL);
  AcDbDictionary *pNamedObj(NULL),*pNameList(NULL);
  AcDbObjectId oid; BOOL doRet(FALSE),unlock(FALSE);
  CString dctName;
  dctName.Format((_T("CGSA_EL_AXES_REACTOR_%d")+ idx));

  if(REACTOR_LOCK == 0){acDocManager->lockDocument(curDoc()); unlock=TRUE;}

  AcDbDatabase *pDb(acdbHostApplicationServices()->workingDatabase());
  if(pDb){
    if(pDb->getNamedObjectsDictionary(pNamedObj, AcDb::kForWrite)==Acad::eOk){
      if(pNamedObj->getAt(dctName, (AcDbObject*&)pNameList, AcDb::kForWrite)==Acad::
    eKeyNotFound){
        pNameList = new AcDbDictionary; pNamedObj->setAt(dctName, pNameList, oid);
      }
      pNamedObj->close();
      if(!oId.isNull() && acdbOpenAcDbEntity(pEnt,oId,AcDb::kForWrite)==Acad::eOk){
        ElectraAxesModifyReactor *pObj(new ElectraAxesModifyReactor()); pObj->eLinkage(
    oId, idx);
        if((pNameList->getAt(_T("AXIS_REACTOR"), oid))==Acad::eKeyNotFound){
          pNameList->setAt(_T("AXIS_REACTOR"), pObj, oid); pObj->close(); pObj=NULL;
        }else{delete pObj; pObj=NULL;}

        if(!oid.isNull()) pEnt->addPersistentReactor(oid); REACTOR_LOCK=0; pEnt->close()
    ; pEnt=NULL; doRet=TRUE;
      }
      pNameList->close();
    }
  }
  if(unlock) acDocManager->unlockDocument(curDoc());
  return(doRet);
}
```

Primer 4.6: Implementacija funkcije addReactorToEntity

Jig

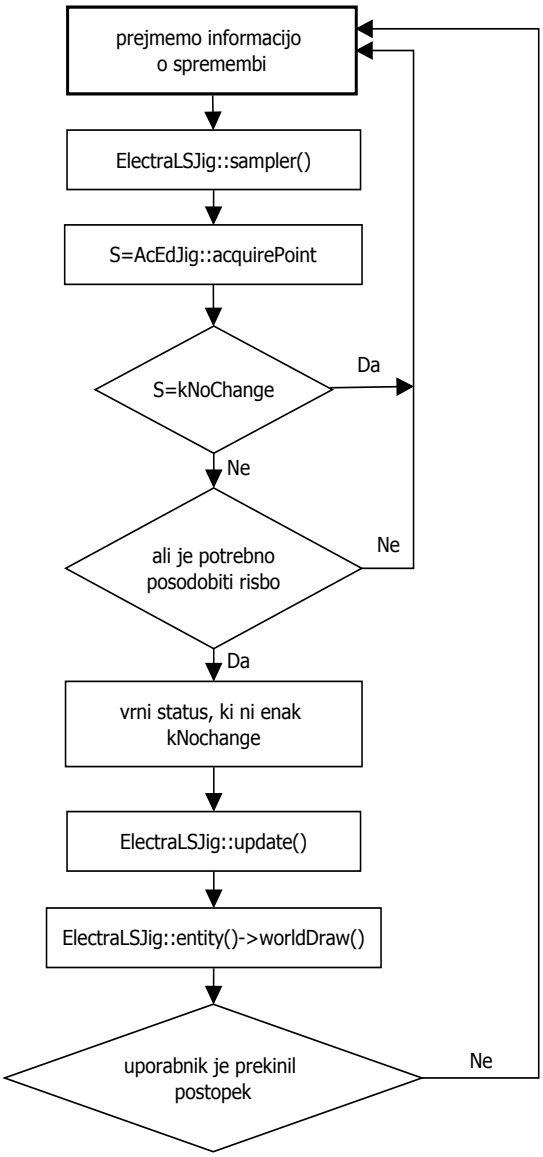
Čeprav že z reaktorji lahko dosežemo kar precejšnjo stopnjo avtomatike in dinamike dela s programom, pa nam CAD platforma omogoča še en korak več. Pri Autodesku so to tehnologijo poimenovali jig. Če smo z reaktorji imeli možnost reagirati ob vsaki spremembi entitete, pa nam jig omogoča, da se odzovemo na vsak premik miškinega kazalca. Na ta način lahko uporabniku programa omogočimo, da s premikanjem miške sproti spremlja, kako se odvisne količine spreminjajo in izrisujejo na zaslonu. Mi smo jig uporabili pri pozicioniranju drogov vzdolž trase. V vzdolžnem pogledu smo tako omogočili, da uporabnik ob zagonu ukaza za pozicioniranje drogov le tega z miško premika po vzdolžnem terenu, pri čemer se vsi definirani vodniki, ki

potekajo preko tega droga, sproti preračunavajo in izrisujejo. Na ta način smo na zaslonu ustvarili neke vrste interaktivno animacijo vzdolžnega poteka drogov in vodnikov, ki jo vodi uporabnik s premikanjem miške. Med obstoječimi rešitvami na trgu nismo zasledili možnosti tovrstnega urejanja, čeprav je za uporabnika zelo intuitiven in enostaven, zato je to vsekakor ena od pomembnih prednosti programa Electra.

Razred programskega vmesnika ObjectARX, ki smo ga uporabili za realizacijo jig-a, je AcEdJig. Da bi implementirali funkcionalen jig, smo morali najprej izpeljati nov razred iz razreda AcEdJig in prekriti sledeče funkcije:

- *AcEdJig::sampler()*
- *AcEdJig::update()*
- *AcEdJig::entity()*

Jig poženemo s klicem metode AcEdJig::drag(). Ta metoda nato ciklično kliče metodo AcEdJig::sampler(). Vsakič, ko v metodi AcEdJig::sampler() zaznamo zadostno spremembo položaja miške, zaključimo funkcijo z vrednostjo AcEdJig::kNormal, sicer pa z AcEdJig::kNoChange. Ciklično klicanje metode prekinemo tako, da funkcijo zaključimo z AcEdJig::kCancel. Če je bila zaznana zadostna sprememba, da moramo posodobiti entitete, ki jih urejamo z vlečenjem miške, potem funkcija AcEdJig::drag() pokliče funkcijo AcEdJig::update(), ki posodobi entitete glede na pridobljene geometrijske vrednosti. Nato se pokliče funkcija AcEdJig::entity(), ki vrne kazalec na entiteto, ki se mora posodobiti na zaslonu in nato še funkcija worldDraw() od same entitete, ki dejansko posodobi izris entitete. Opisani cikel se ponavlja, vse dokler uporabnik ne prekine postopka s klikom miške ali tipkovnico. Namen funkcije AcEdJig::sampler() je, da pridobi geometrijske podatke na podlagi uporabnikovega premikanja miške in jih interpretira kot razdaljo, kot ali točko [4]. Na sliki 4.2 je prikazan diagram poteka za jig. Za primer urejanja položaja drogov nadzemnega elektroenergetskega voda smo iz razreda AcEdJig izpeljali razred ElectraLSJig in prekrili vse potrebne funkcije. Dodali smo funkcijo ElectraLSJig::doIt(), s katero zaženemo urejanje z Jig-om,



Slika 4.2: Jig - diagram poteka [4]

pred tem pa še inicializiramo vse potrebne parametre za izračun poteka vodnikov. V primeru iz dodatka B.1 je za vsako prekrito funkcijo opisano, katere funkcionalnosti smo vnjej implementirali.

4.2 Podatkovni sloj

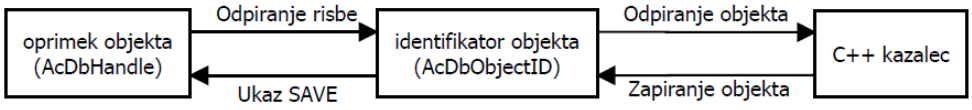
Podatkovni sloj programa smo realizirali tako, da se vsi podatki shranjujejo v bazo DWG ali brišejo iz nje sproti, ko so ustvarjeni, spremenjeni ali izbrisani. Ko uporabnik shrani risbo DWG, se skupaj z njo na disk zapišejo tudi vsi podatki daljnovoda, ki so bili ustvarjeni z našim programom za načrtovanje nadzemnih elektroenergetskih vodov. Ko uporabnik naslednjič risbo odpre v programu CAD, lahko nadaljuje z delom. Za shranjevanje v bazo DWG smo uporabili razrede in mehanizme, ki jih ponuja programski vmesnik ObjectARX in sicer razred AcDbEntity, XDATA, Slovarje ter razred XRecord.

4.2.1 Uporabljeni načini shranjevanja podatkov v DWG

Entiteta (AcDbEntity)

Najbolj osnoven in najpogosteje uporabljen način shranjevanja v bazo DWG je sam izris entitete v risbo. Entitete so predstavljene z razredi, izpeljanimi iz razreda AcDbEntity, ki je izpeljava razreda AcDbObject. AcDbObject je osnovni razred za vse objekte baze DWG, AcDbEntity pa je osnovni razred za vse objekte, ki imajo tudi grafično upodobitev. Poleg tega da entiteta v risbi predstavlja grafično upodobitev nekega podatka, lahko predstavlja tudi podatek sam, saj s svojo geometrijo nosi informacijo. Na tak način v našem programu shranjujemo podatek o geometriji trase, kar pomeni da je polilinija, ki je uporabljena za izris trase, hkrati tudi nosilec informacije o njenem poteku. Natančnejši postopek vnosa entitete v bazo DWG smo si pogledali na primeru izrisa linije 4.2. Do objektov in posledično tudi do entitet v bazi DWG lahko dostopamo na tri različne načine, in sicer preko oprimka (angl. handle), identifikatorja ali C++ kazalca. Ko program Autocad ni v teku, je

risba DWG shranjena v datotečnem sistemu in vsi objekti, vsebovani v risbi, so določeni z njihovimi oprimki. Ko risbo odpremo v programu Autocad, pa dobijo vsi objekti unikaten identifikator v bazi. Z uporabo tega identifikatorja lahko posamezni objekt odpremo za branje ali pisanje in takrat nam sistem vrne C++ kazalec na odprt objekt, ki je veljaven, vse dokler objekta ne zapremo s funkcijo AcDbObject::close(). Vsak objekt, ki ga odpremo, moramo čimprej po uporabi zapreti.



Slika 4.3: Dostopanje do objektov v bazi DWG [4]

Oprimek je v programskem vmesniku ObjectARX predstavljen z razredom AcDbHandle, identifikator pa z razredom AcDbObjectId. Funkcije programskega vmesnika, ki nam omogočajo dostopanje in upravljanje z objekti v različnih situacijah so:

- *getAcDbObjectId* za pridobivanje identifikatorja objekta iz njegovega oprimka AcDbHandle [4]
- *acdbOpenObject* za pridobivanje C++ kazalca na objekt v odprti bazi DWG, če imamo poznan identifikator objekta AcDbObjectId [4]
- *AcDbObject::getAcDbHandle* za pridobivanje oprimka iz objekta, če imamo poznan C++ kazalec na odprt objekt [4]

XDATA

Kot smo videli, lahko za shranjevanje geometrijskih podatkov velikokrat uporabimo kar entiteto samo. Vendar pa velikokrat obstaja potreba, da bi določen podatek, ki ima svojo grafično upodobitev v risbi, obogatili še

s kakšnimi drugimi podatki. XDATA je eden od mehanizmov shranjevanja podatkov v samo AutoCAD entiteto. Ta način ima kar nekaj omejitev, in sicer ena glavnih omejitev je, da je zgornja meja količine podatkov na eno entiteto omejena na 16 kilobajtov, po drugi strani pa je pomnilniška izkoriščenost dobra, kar pomeni, da je mehanizem XDATA najbolj primeren za shranjevanje manj obsežnih podatkov. Razred AcDbObject vsebuje funkciji setXData(const resbuf* xdata) in xData(const char* regappName = NULL) za vpisovanje in branje verige resbuf s podatki v ali iz objekta. Resbuf je struktura, ki se v Autocadu in ObjectARX-u uporablja za vsebovanje podatkov v obliki seznama (glej 4.7).

```
union ads_u_val {
 ads_real rreal;
 ads_real rpoint[3];
 short rint; // Must be declared short, not int.
 char *rstring;
 long rlname[2];
 long rlong;
 struct ads_binary rbinary;
};
struct resbuf {
 struct resbuf *rbnext; // Linked list pointer
 short restype;
 union ads_u_val resval;
};
```

Primer 4.7: Definicija result-buffer strukture [4]

Mi smo XDATA uporabili v sledeče namene:

- Za shranjevanje imena osi v linijo droga in linijo vrvi. Ime osi namreč enolično določa traso. Tako za vsak drog in vsako vrv lahko določimo, kateri trasi pripada.

- Za shranjevanje unikatne identifikacijske številke droga v linijo droga. Ob izbiri linije droga lahko na ta način iz baze drogov preberemo zapis z vsemi lastnostmi droga.

- Za shranjevanje unikatne identifikacijske številke vrvi v polilinijo vrvi. Ob izbiri polilinije vrvi lahko na ta način iz baze vrvi preberemo zapis z vsemi lastnostmi vrvi.

- Za shranjevanje številke oprimka linije droga v pripadajoče entitete tekstov, ki predstavljajo opise drogov. Ob izbiri, premikanju ali brisanju določenega teksta tako vemo, kateri poliliniji droga pripada tekst in tako lahko linijo droga vključimo v izbiro, premikanje ali brisanje.

- Za shranjevanje identifikacijskega niza izrisanega vzdolžnega profila v vse entitete, ki so izrisane v okviru vzdolžnega profila. Na ta način lahko naredimo izbor vseh entitet, ki pripadajo določenemu vzdolžnemu profilu. To nam pride prav npr. pri brisanju celotnega profila.

Slovar (AcDbDictionary)

Kot glavne vsebovalnike za shranjevanje podatkov, ki jih uporablja program, smo uporabili slovarje. Slovar je v programskem vmesniku ObjectARX realiziran z razredom AcDbDictionary [4]. Z njim lahko implementiramo objekte, ki so prisotni v bazi DWG in so namenjeni za mapiranje tekstovnih ključev z objekti, ki so shranjeni v bazi DWG. Vsak vnos v AcDbDictiorary je torej unikaten in je sestavljen iz unikatnega objekta AcDbObject in tekstovnega niza, ki predstavlja ključ zapisa. Mi smo za naše potrebe hranjenja podatkov tvorili zapise z objekti tipa AcDbXrecord.

XRecord

XRecord nam podobno kot XDATA omogoča, da v DWG dodajamo podatke, specifične za aplikacijo. Za razliko od XDATA, XRecord nima omejitve v količini shranjenih podatkov. XRecord je lahko v lasti poljubnega objekta vključno s slovarjem (AcDbDictionary). XRecord je realiziran z uporabo razreda AcDbXrecord, ki je podrazred razreda AcDbObject. Razred AcDbXRecord vsebuje dve funkciji za vpisovanje in branje podatkov v obliki verige resbuf (glej 4.7), in sicer setfromRbChain() in rbChain() [4]. Mi smo XRecord uporabili za implementacijo zapisov v slovarjih AcDbDictionary.

4.2.2 Implementacija podatkovnega sloja

Vse zgoraj opisane načine shranjevanja podatkov, ki nam jih ponuja programski vmesnik ObjectARX, smo uporabili pri implementaciji podatkovnega sloja. V ta namen smo implementirali razrede, ki nam služijo za upravljanje s podatki daljnovoda. Elemente daljnovoda smo v programski podatkovni strukturi razdelili na tri osnovne razrede:

- Razred *ElectraPole* vsebuje lastnosti in implementira funkcije za reprezentacijo droga daljnovoda. Lastnosti, ki jih vsebuje razred ElectraPole so: unikatni identifikator droga, ime droga, stacionaža droga, tip droga (zatezni, kotni ali nosilni), niz tipov obešanja, niz tipov konzol, niz tipov izolatorjev, višina droga, oprimek entitete za prikazovanje droga v vzdolžnem profilu, oprimek entitete za prikazovanje droga na trasi ter ime simbola za prikaz v trasnem načrtu. Pomembnejše funkcije razreda ElectraPole so poleg tistih za nastavljanje in branje lastnosti razreda še:
  - DrawPoleLineLS za izris linije droga v vzdolžnem profilu
  - DrawPoleSignAX za izris oznake droga v tlorisnem pogledu trase
  - UpdatePoleLineLS za posodobitev linije droga v vzdolžnem rofilu. (Uporablja se znotraj Jig-a)
  - UpdatePoleSignAX za posodobitev simbola droga v v tlorisnem pogledu trase. (Uporablja se znotraj Jig-a)
  - DrawPoleTextsLS za izpis opisov nad drogom v vzdolžnem profilu
  - WriteEED_PoleLS za vpis identifikatorja droga in imena trase v XDATA linije droga v vzdolžnem profilu
  - WriteEED_PoleAX za vpis identifikatorja droga in imena trase v XDATA simbola v tlorisnem pogledu
  - ErasePoleLineLS, ErasePoleLineAX in ErasePoleTextsLS za brisanje linije droga v vzdolžnem, brisanje simbola droga v tlorisnem pogledu trase ter brisanje opisov nad drogom v vzdolžnem profilu.
- Razred *ElectraCable* vsebuje lastnosti in implementira funkcije za re-

prezentacijo električnega vodnika daljnovoda. Lastnosti, ki jih vsebije razred ElectraCable so:unikatni identifikator vodnika, ime vodnika, oprimek entitete za prikazovanje vodnika v vzdolžnem profilu, oprimek entitete za prikazovanje varnostne razdalje od vodnika, identifikator uporabljene definicije ElectraCatenary za izračun povesne verižnice, identifikator droga, na katerem se začne ta vodnik ter varnostna razdalja, ki je definirana za ta vodnik. Pomembnejše funkcije razreda ElectraCable so poleg tistih za nastavljanje in branje lastnosti razreda še:

  - DrawCablePoly za izris polilinije vodnika ter polilinije varnostne razdalje v obliki povesne verižnice v vzdolžnem profilu
  - UpdateCablePoly za posodobitev polilinije vodnika ter polilinije varnostne razdalje v vzdolžnem profilu (Uporablja se znotraj Jiga)
  - WriteEED za vpis identifikatorja vodnika in imena trase v XDATA polilinije vodnika ter polilinije za varnostno razdaljo v vzdolžnem profilu
  - EraseCablePoly za brisanje polilinije vodnika in polilinije varnostne razdalje v vzdolžnem profilu

- *ElectraCatenary* vsebuje lastnosti in definicije, ki vplivajo na izračun povesne verižnice. Skupek lastnosti, ki jih vsebuje ta razred, je tudi ena izmed lastnosti razreda ElectraCable, saj v veliki meri definira parametre povesne verižnice, ki jo zavzame električni vodnik med dvema drogovoma. Lastnosti, ki jih vsebuje razred ElectraCatenary, so: unikatni identifikator definicije lastnosti verižnice, ime definicije lastnosti verižnice, tip vodnika, maksimalna natezna napetost vodnika, osnovna temperatura ter faktor dodatnega bremena.

Objekti, ki jih tvorimo s pomočjo opisanih treh razredov ElectraPole, ElectraCable in ElectraCatenary, so glavni gradniki našega modela nadzemnega elektroenergetskega voda in se uporabljajo pri skoraj vseh izračunih in operacijah, ki jih naš program omogoča. Zaradi potrebe samega hranjenja teh

objektov v bazi ter lažjih poizvedb in posodobitev objektov smo se odločili, da bomo implementirali vsebovalne razrede. Naloga takega vsebovalnega razreda je, da implementira shranjevanje objektov v bazo DWG, učinkovito dostopanje do objektov ter enostavno posodabljanje baze na način, da stanje v bazi v vsakem trenutku ustreza trenutnemu stanju daljnovoda v projektu. Za vsakega od treh osnovnih tipov objektov smo naredili svoj vsebovalni razred, in sicer StoragePoles, StorageCables in StorageCatenaries. Vsak od teh treh vsebovalnih razredov vsebuje svoj seznam objektov ustreznega tipa in implementira naslednje osnovne funkcije za poizvedovanje ter upravljanje s seznamom objektov:

- Init je funkcija za inicializiranje vsebovalnega razreda. Ob inicializaciji se za podano traso preberejo vse vrednosti iz baze DWG v objekt ustreznega vsebovalnega razreda.

- Count je funkcija, ki vrne število elementov v seznamu vsebovalnega razreda.

- Update je funkcija vsebovalnega razreda, ki nam omogoča posodabljanje vrednosti nekega elementa v seznamu ali pa dodajanje novega, če element s podanim identifikatorjem še ne obstaja v seznamu. Hkrati s seznamom v objektu se posodobi tudi zapis v bazi DWG, natančneje v ustreznem XRecord-u.

- Delete je funkcija vsebovalnega razreda, s katero izbrišemo element s podanim identifikatorjem iz seznama in iz baze DWG.

- GetAt je funkcija vsebovalnega razreda, ki nam omogoča dostopanje do elementa, ki se nahaja v seznamu na podanem indeksu.

- GetById je funkcija vsebovalnega razreda, ki nam omogoča dostopanje do elementa s podanim identifikatorjem.

Poglejmo si še specifike vsakega od treh vsebovalnih razredov posebej:

- *StoragePoles* je razred, ki vsebuje urejen seznam objektov tipa ElectraPole. Seznam je urejen po naraščajočih stacionažah drogov. Za shranjevanje seznama drogov uporablja XRecord z imenom CGSA_E_POLES_{*Ime trase*}. Razred StoragePoles implementira poleg že opisani funkcij še:

  - GetNextPole in GetPrevPole sta funkciji vsebovalnega razreda, ki omogočata dostopanje do droga iz seznama drogov, ki je naslednji ali predhodni po stacionaži glede na drog s podanim identifikatorjem.
  - GetNextTensionPole in GetPrevTensionPole sta funkciji vsebovalnega razreda, ki omogočata dostopanje do droga iz seznama drogov, ki je prvi naslednji ali predhodni zatezni drog po stacionaži glede na drog s podanim identifikatorjem.
  - UpdateAllCables je funkcija, ki nam omogoča, da posodobimo pozicije obešanj definiranih vodnikov preko celotnega seznama drogov. S klicem te funkcije zagotovimo, da so definirani vodniki ves čas v pravilnem zaporedju vpeti na drogove.

- *StorageCables* je razred, ki vsebuje seznam objektov tipa ElectraCable. Razred implementira zgoraj opisane funkcije za poizvedovanje in upravljanje s seznamom vodnikov. Za shranjevanje seznama vodnikov uporablja XRecord z imenom CGSA_E_CABLES_{*Ime trase*}

- *StorageCatenaries* je razred, ki vsebuje seznam objektov tipa ElectraCatenary. Za shranjevanje seznama definicij parametrov verižnice uporablja XRecord z imenom CGSA_E_CATENARIES_{*Ime trase*}. Razred StorageCatenaries implementira poleg že opisani funkcij še:

  - DeleteAllWithCatenary je funkcija, ki nam omogoča brisanje vseh vrvi, ki imajo kot svojo lastnost definiran skupek lastnosti verižnice s podanim identifikatorjem.

4.3 Numerični izračuni

Glavnino numeričnih izračunov v programu smo morali opraviti pri izračunih povesne verižnice ter pri izračunih nateznih napetosti. Pri izračunih smo dali glavni poudarek temu, da so izračuni v skladu s standardi in uveljavljeno teorijo nadzemnih elektroenergetskih vodov. Predvsem pri računanju povesne verižnice smo morali poleg pravilnosti paziti tudi na hitrost izračuna, saj se mora pri urejanju daljnovoda z uporabo jig-a povesna verižnica izračunavati sproti, medtem ko z miško vlečemo drog vzdolž trase.

4.3.1 Razred za izračun povesne verižnice

Za potrebe izračunavanja poteka povesne verižnice za posamezne vodnike smo implementirali razred CableCalculation. Razred implementira naslednje funkcije:

- *CalculateCable* je vrhnja funkcija razreda, ki jo pokličemo, kadar hočemo izračunati potek določenega vodnika. V funkcijo kot vhodni parameter podamo vodnik v obliki objekta tipa ElectraCable, v izhodnem parametru pa dobimo tabelo točk, ki tvorijo verižnico, ki jo zavzame vodnik.

- *CalculateHangingPoints* je funkcija, ki za vodnik, podan kot vhodni parameter tipa ElectraCable, izračuna koordinate točk obešanja na drogovih. Te točke se nato uporabijo kot vhodni parameter funkcije Calc.

- *Calc* je osrednja funkcija izračuna. Za vhodne parametre vzame objekt tipa ElectraCatenary, objekt tipa BasicCable ter množico točk obešanja, ki smo jih predhodno izračunali s funkcijo CalculateHangingPoints. V izhodnem parametru funkcija vrne množico točk, ki tvorijo izračunano povesno verižnico, potekajočo skozi točke obešanja.

- *CalcIdealSpan* je funkcija, ki izračuna in vrne idealno razpetino. Kot

vhodni parameter ji podamo množico točk obešanja. Funkcija se uporablja v okviru izračunov v funkciji Calc.

- *CalcSigma* je funkcija, ki izračuna in vrne natezno napetost vodnika pri določeni temperaturi. Natezno napetost izračunamo iz položajne enačbe po Newtnovem iterativnem postopku. Funkcija se uporablja v okviru izračunov v funkciji Calc.

- *CalcCatenary* je funkcija, s katero na podlagi vhodne natezne napetosti vodnika ter maksimalnega povesa izračunamo potek verižnice med dvema obesiščema. Tudi ta funkcija se uporablja v okviru izračunov v funkciji Calc.

4.3.2 Teoretične osnove za izračun povesne verižnice

Za realizacijo vseh izračunov, ki jih opravimo v razredu CableCalculation, smo morali najprej pridobiti nekatere teoretične osnove za izračun povesne verižnice. Namen je bil iz ustrezne literature izluščiti formule potrebne za realizacijo numeričnih izračunov, ki jih potrebujemo v programu, ter dobiti osnovni vpogled v teorijo nadzemnih elektroenergetskih vodov. V globine in podrobnosti te teorije se nismo spuščali, saj to ni bil osnovni cilj tega dela. V nadaljevanju si poglejmo katere teoretične zaključke, povzete iz literature [1, 2, 3], smo uporabili v naših izračunih.

Povesna verižnica je krivulja, ki predstavlja potek v dveh točkah vpetega vodnika. Vodnik je vpet med dva zatezna ali kotna drogova. Pri izračunu verižnice predpostavljamo, da so vrvi idealno upogljive tako kot verige.

Zatezno polje, imenovano tudi napenjalno polje, je del nadzemnega voda med dvema zateznima ali kotnima drogovoma.

Razpetina je definirana kot horizontalna razdalja med dvema drogovoma, oziroma med dvema obesiščema, če smo natančnejši.

Idealna razpetina je definirana kot uniformna razpetina, ki na nekem zateznem polju najbolje opiše vse razpetine tega zateznega polja. Računamo jo po formuli 4.1

$$s_i = \sqrt{\frac{\sum_j s_j^3}{\sum_j s_j}} \tag{4.1}$$

kjer sta:

- s_i idelana razpetina verižnice in
- s_j razpetina med dvema drogovoma ustrezne verižnice

Dodatno zimsko breme izrazimo kot povečano specifično težo vodnika po formuli 4.2. Pri nas je dodatno zimsko breme enako največjemu dodatnemu zimskemu bremenu, ki se na določenem mestu pojavlja vsakih pet let.

$$\Delta p = \frac{0.18\sqrt{d_v}}{A}\left[\frac{daN}{m \cdot mm^2}\right] \tag{4.2}$$

kjer so:

- Δp specifična teža dodatnega zimskega bremena (daN/m),
- d_v premer vrvi (mm) in
- A presek vrvi (mm^2).

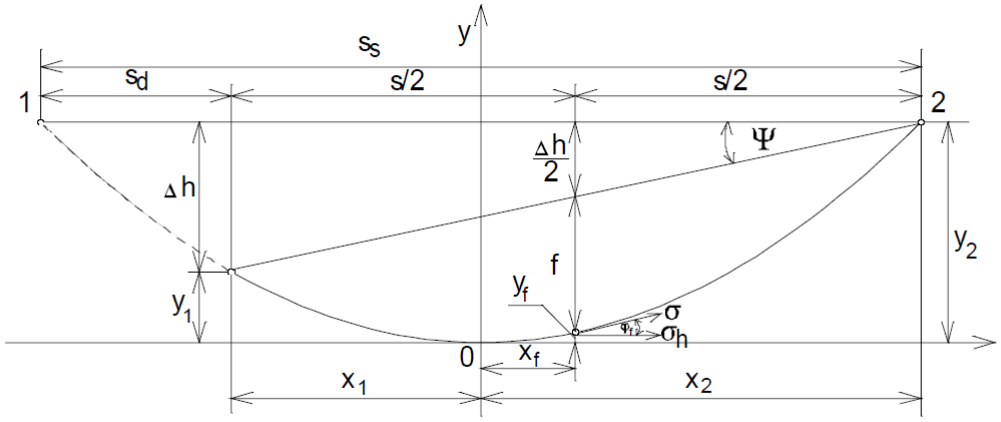
Pri računanju dodatnega zimskega bremena se upošteva tudi faktor dodatnega zimskega bremena (g), ki ima lahko naslednje vrednosti: 1.0, 1.6, 2.5 ali 4.0. Z upoštevanje tega faktorja pridemo do **skupne specifične teže vrvi**, ki jo računamo po formuli 4.3

$$\bar{p} = p + g\Delta p\left[\frac{daN}{m \cdot mm^2}\right] \tag{4.3}$$

kjer so:

- $\bar{p}$ skupna specifična teža vrvi,

- p osnovna specifična teža vrvi,
- g faktor dodatnega zimskega bremena in
- Δp specifična teža dodatnega zimskega bremena.



Slika 4.4: Verižnica z različnimi višinami obesišč

Pri izračunu verižnice in nateznih sil sta pomembna še dva pojma, ki predstavljata merilo pri izbiri značilnega temperaturnega stanja, ko nastopi največja natezna napetost oziroma največji poves. Ta dva pojma sta kritična razpetina in kritična temperatura.

Kritična razpetina s_k je tista razpetina, pri kateri je natezna napetost pri $-5°C$ z dodatnim bremenom σ_{-5+db} natanko enaka natezni napetosti pri $-20°C$ brez dodatnega bremena σ_{-20}. Kritično razpetino računamo po formuli 4.4

$$s_k = \sigma_{max}\sqrt{\frac{360\alpha}{\bar{p}^2 - p^2}[m]} \tag{4.4}$$

kjer so:

- s_k kritična razpetina verižnice,

- σ_{max} dopustna napetost vrv,
- α temperaturni koeficient vrvi,
- $\bar{p}$ skupna specifična teža vrvi in
- p osnovna specifična teža vrvi

Če je dejanska razpetina večja od kritične razpetine, potem nastopi največja natezna napetost pri $-5°C$ z dodatnim bremenom. Če pa je dejanska razpetina manjša od kritične razpetine, potem nastopi največja natezna napetost pri $-20°C$. Mi smo pri tem preverjanju za dejansko razpetino vzeli idealno razpetino s_i.

Kritična temperatura ϑ_k je tista pozitivna temperatura, pri kateri je poves natanko enak povesu pri $-5°C$ z dodatnim bremenom f_{-5+db}. Računamo jo po formuli 4.5.

$$\vartheta_k = \frac{\sigma_{max}}{E\alpha}\left(1 - \frac{p}{\bar{p}}\right) - 5[°C] \tag{4.5}$$

kjer so:

- ϑ kritična temperatura okolice,
- σ_{max} dopustna napetost vrv,
- E modul elastičnosti vrvi,
- α temperaturni koeficient vrvi,
- $\bar{p}$ skupna specifična teža vrvi in
- p osnovna specifična teža vrvi

Če je kritična temperatura manjša od $40°C$, potem nastopi maksimalni poves f_{max} pri temperaturi $40°C$, v nasprotnem primeru pa pri temperaturi $-5°C$ in dodatnem zimskem bremenu.

S predpostavko, da je koordinatni sistem v temenu verižnice, lahko verižnico računamo po formuli 4.6.

$$y = \frac{\sigma}{\bar{p}} ch \frac{\bar{p}x}{\sigma} - \frac{\sigma}{\bar{p}} \tag{4.6}$$

Klasična položajna enačba za različne višine obesišč ima obliko 4.7. Osnovni položaj je fiksiran s parametri $\vartheta_0, \sigma_{0h}, p_0$.

$$\frac{p^2 s^2}{24\sigma^2} - \frac{\bar{p}^2 s^2}{24\sigma_{max}^2} = \alpha(\vartheta - \vartheta_0) + \frac{\sigma - \sigma_{max}}{E cos\Psi} \tag{4.7}$$

kjer so:

- σ napetost vrvi pri temperaturi okolice ϑ,
- ϑ temperatura okolice, ki zajema vrednosti od $-20°C$ do $40°C$ v presledkih po $5°C$,
- ϑ_0 osnovna temperatura okolice,
- σ_{max} dopustna napetost vrvi,
- E modul elastičnosti vrvi,
- α temperaturni koeficient vrvi,
- $\bar{p}$ skupna specifična teža vrvi,
- p osnovna specifična teža vrvi in
- Ψ kot med točkama vpenjanja vrvi in horizontalo

Ob enem od navedenih osnovnih stanj lahko po položajni enačbi za vsako temperaturo v predpisanem območju od $-20°C$ do $40°C$ izražunamo ustrezno horizontalno natezno napetost.

Poleg nateznih napetosti nas zanimajo tudi povesi v odvisnosti od temperature. Določimo jih lahko z **osnovno enačbo za poves** pri različnih višinah obesišč, ki ima obliko 4.8

$$f = \frac{ps^2}{8\sigma_h cos\Psi} + \frac{p^3 s^4}{384\sigma_h^3 cos\Psi} \tag{4.8}$$

Čeprav je sicer položajna enačba 4.7 ekzaktno rešljiva, smo mi v programu uporabili numerično reševanje, in sicer s pomočjo Newtonovega iterativnega postopka. Za uporabo tega postopka smo enačbo 4.7 nekoliko preoblikovali in dobili enostavno obliko položajne enčbe

$$\sigma_h + m = \left(\frac{n}{\sigma_h}\right)^2 \tag{4.9}$$

kjer velja

$$m = \frac{p_0 s^2}{24\sigma_{0h}^2} E cos\Psi + \alpha(\vartheta - \vartheta_0) E cos\Psi - \sigma_{0h}, \tag{4.10}$$

$$n = ps\sqrt{\frac{E cos\Psi}{24}} \tag{4.11}$$

.

Parametra m in n lahko po formulah 4.10 enostavno izračunamo, saj imamo vse potrebne podatke. Na podlagi teh parametrov lahko potem s pomočjo Newtonovega iterativnega postopka izračunamo horizontalno natezno napetost σ_h.

Funkcijo, ki izračuna množico točk, ki predstavljajo potek povesne verižnice na konkretni razpetini pri določeni horizontalni natezni napetosti ter specifični teži vodnika, smo implementirali po naslednjem postopku:

1. Iz podanih koordinat točk obesišč za določeno razpetino izračunamo horizontalno dolžino razpetine $s = x_{ob2} - x_{ob1}$.

2. Iz podanih koordinat točk obesišč za določeno razpetino izračunamo višinsko razliko obesišč $\Delta h = y_{ob2} - y_{ob1}$

3. Iz podane horizontalne natezne napetosti σ_h ter specifične teže vodnika p izračunamo parameter verižnice $a = \frac{\sigma_h}{p}$

4. Izračunamo fiktivni dodatek razpetine,
$$s_d = asinh\left(\frac{\Delta h}{sinh(\frac{s}{2a})2a}\right)2a$$
ki bi bil potreben da dopolnimo verižnico do enakih višin obesišč.

5. Izračunamo
$$x_1 = \frac{s - s_d}{2} \quad in \quad x_2 = \frac{s + s_d}{2}$$
(glej sliko 4.4)

6.
$$s_s = s + s_d$$

7.
$$f = a\left(cosh\frac{s_s}{2a} - 1\right)$$

8. Celotno razpetino razdelimo na n delov $\Delta b = \frac{s_s}{n}$

9. Če je $\Delta h > 0$, postavimo začetno vrednost iteracijskega intervala $b_{zac} = -x_{ob1}$ in konec iteracijskega intervala $b_{kon} = x_{ob2}$. V nasprotnem primeru pa postavimo začetno vrednost iteracijskega intervala $b_{zac} = -x_{ob2}$ in konec iteracijskega intervala $b_{kon} = x_{ob1}$.

10. Nato po korakih Δb povečujemo vrednost iteratorja b od vrednosti b_{zac} do b_{kon}

11. V vsakem koraku izračunamo: $f_b = a * (cosh\frac{b}{a} - 1) - f$ ter koordinati točke verižnice:
$$X = \begin{cases} x_{ob1} + \frac{s_s}{2} + b - s_d, & \text{če velja } \Delta h > 0 \\ x_{ob1} + \frac{s_s}{2} + b, & \text{če velja } \Delta h <= 0 \end{cases}$$

$$Y = \begin{cases} y_{ob1} + f_b + \Delta h, & \text{če velja } \Delta h > 0 \\ y_{ob1} + f_b, & \text{če velja } \Delta h <= 0 \end{cases}$$

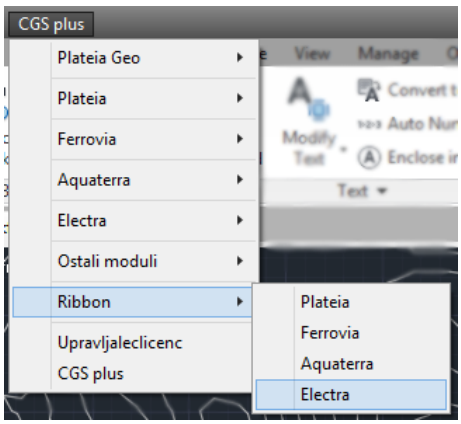
4.4 Tipičen potek uporabe programa

V tem poglavju bomo na enostavnem primeru pokazali uporabo programa za načrtovanje nadzemnih elektroenergetskih vodov, predstavili uporabniški vmesnik ter grafične rezultate programa. Ker je program integriran v programsko opremo CGS Infrastructure Design Suite in se poganja v okolju Autocad, nekateri elementi grafičnega vmesnika niso del obravnavanega programa. Za opisani primer smo uporabili risbo DWG s predpripravljeno terensko površino.

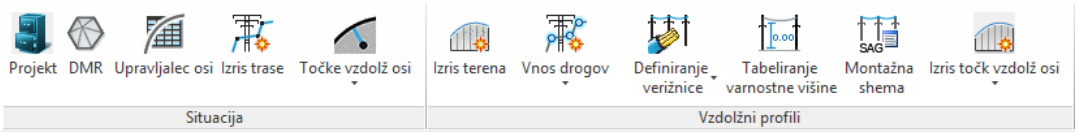
4.4.1 Priprava okolja in kreiranje projekta

Po zagonu programa Autocad z nameščeno programsko opremo CGS Infrastructure Design Suite najprej odpremo datoteko DWG, ki vsebuje ustrezno terensko površino, po kateri bomo speljali traso nadzemnega elektroenergetskega voda. Med Autocad-ovimi meniji lahko opazimo meni "CGS plus", pod katerim so zbrani podmeniji za dostop do vseh funkcionalnosti programske opreme CGS Infrastructure Design Suite. Med njimi je tudi programski modul Electra, ki ima svoj set menijev z vsemi ukazi in orodji, na voljo pa imamo tudi poenostavljen nabor najpomembnejših ukazov v obliki preglednega orodnega traku (slika 4.6) (angl. ribbon), ki ga vklopimo z ukazom iz menija kot je prikazano na sliki 4.5.

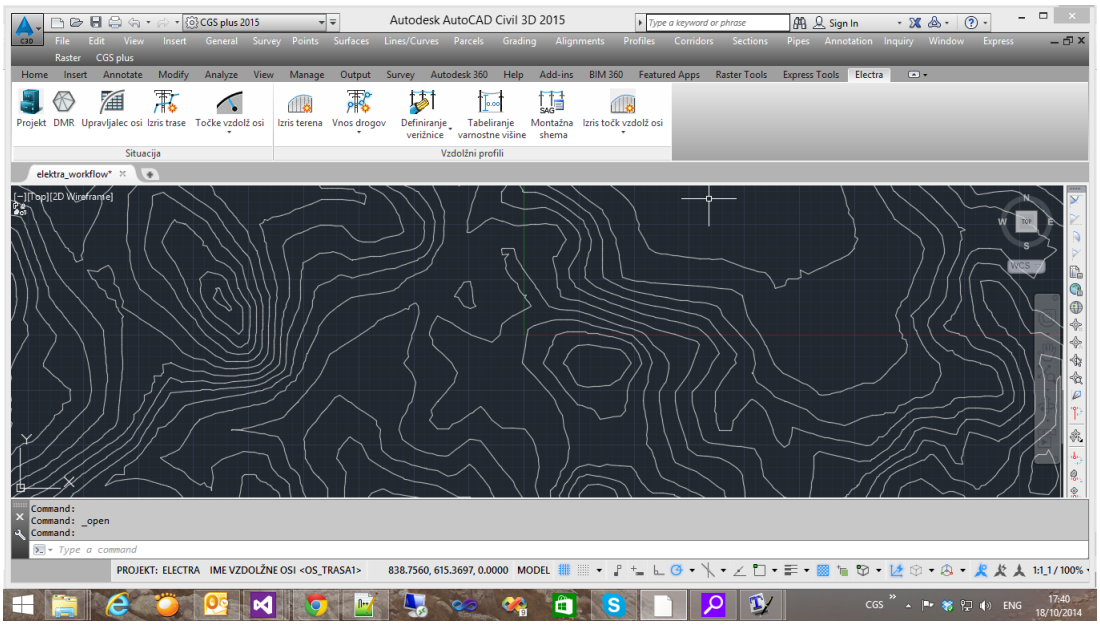
Po vklopu orodnega traku imamo vzpostavljeno začetno okolje (slika 4.7) in lahko začnemo z načrtovanjem nadzemnega elektroenergetskega voda. Pravila uporabe programske opreme CGS Infrastructure Design Suite od nas zahtevajo, da vedno delamo s programom v okviru nekega projekta, zato z orodnega traku najprej izberemo ukaz "Projekt"ter ustvarimo nov projekt in



Slika 4.5: Vklop orodnega traku

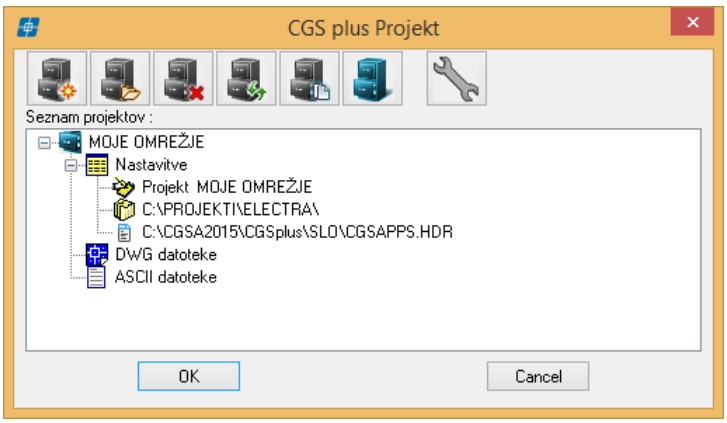


Slika 4.6: Orodni trak



Slika 4.7: Electra z odprto terensko površino

ga poimenujemo npr. MOJE OMREŽJE (slika 4.8). V naslednjem koraku z

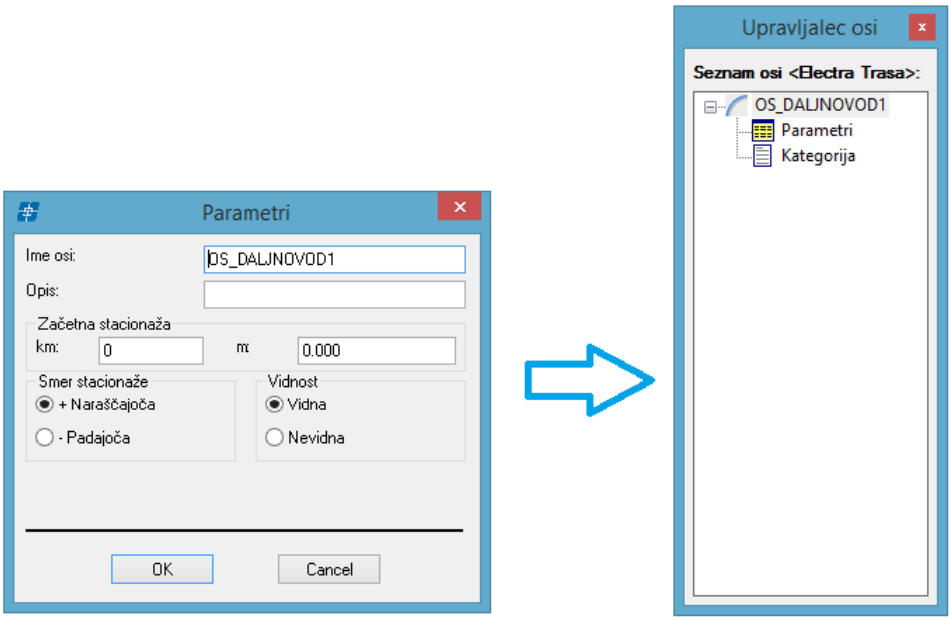


Slika 4.8: Kreiranje projekta

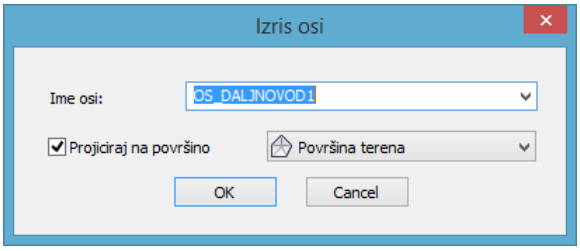
orodnega traku izberemo ukaz "Upravljalec osi"in ustvarimo novo os, ki bo predstavljala našo traso. Poimenujmo jo OS_DALJNOVOD1. (slika 4.9)

4.4.2 Vnos tlorisnega poteka trase

Sedaj imamo vse pripravljeno za vnos tlorisa poteka trase v našo risbo. To storimo z izbiro ukaza "Izris trase" z orodnega traku. Odpre se nam pogovorno okno (na sliki 4.10), v katerem izberemo os in terensko površino, na katero bi radi projicirali potek trase. Ker imamo v našem primeru le eno os in eno terensko površino, potrdimo privzete vrednosti in nadaljujemo z vnosom trase daljnovoda. V ukazni vrstici se pojavi vmesnik za vnos polilinije trase (slika 4.11), ki nam omogoča vnos zaporednih točk polilinije, kakor tudi vnos že obstoječe polilinije ali pa kombinacijo obeh možnosti. V našem primeru nimamo v risbi nobene obstoječe polilinije, zato vnesemo novo traso z zaporednim klikanjem točk v risbi (slika 4.12). Ko vnesemo celotno traso, vnos zaključimo s pritiskom na tipko K oziroma Enter, nakar nam program zgenerira traso, jo projicira na izbrano terensko površino ter samodejno postavi zatezna drogova na začetku in na koncu trase ter kotne drogove na vsaki točki preloma trase. Rezultat lahko vidimo na sliki 4.13. Zatezni



Slika 4.9: Kreiranje nove osi ter pojavitev le te v upravljalcu osi



Slika 4.10: Izbira osi in terenske površine za izris trase.

Slika 4.11: Vmesnik ukazne vrstice za izris trase

Slika 4.12: Vnos trase s klikanjem točk

Slika 4.13: Trasa daljnovoda v tlorisnem pogledu

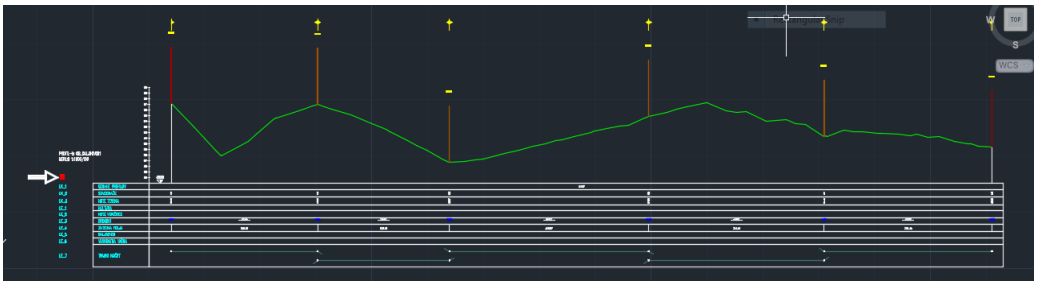
(a) Zatezni drog (b) Nosilni drog (c) Kotni drog

Slika 4.14: Trije tipi drogov

drogovi so na poliliniji trase označeni z rdečimi krogci, kotni drogovi pa z rjavimi krogci. Kasneje bomo na traso vnašali tudi nosilne drogove, ki pa so označeni z modrimi krogci (slika 4.14). Poudariti je potrebno, da imajo tudi kotni drogovi funkcijo zateznih drogov. Od navadnih zateznih drogov se razlikujejo le po tem, da jih ni mogoče poljubno premikati vzdolž trase, temveč so vedno postavljeni v točki, kjer trasa spremeni smer.

4.4.3 Generiranje vzdolžnega poteka trase

Traso smo z določitvijo tlorisnega poteka deloma že umestili v prostor, vendar pa nam za dokončno umestitev manjka še izdelava vzdolžnega poteka. Omenili smo že, da je program ob izdelavi trase le to že tudi samodejno projiciral na terensko površino. To pomeni, da imamo izračunan višinski potek terena vzdolž trase, kar nam pomeni osnovo za izdelavo vzdolžnega poteka trase. Izdelavo vzdolžnega poteka trase bomo začeli z izbiro ukaza 'Izris terena' z orodnega traku. Pojavi se enotno pogovorno okno programske opreme CGS Infrastructure Design Suite za izris terena v vzdolžni profil, kjer lahko nastavimo nekaj osnovnih parametrov izrisa vključno z merilom izrisa, vendar se v podrobnosti na tem mestu ne bomo spuščali. Po potrditvi parametrov nam program ponudi, da s klikom v risbo določimo mesto izrisa vzdolžnega poteka in na tem mestu se pojavi izris, kot ga vidimo na sliki 4.15. Zelena polilinija prikazuje višinski potek terena vzdolž trase, na tem poteku so prikazani zatezni in kotni drogovi, ki so se samodejno vnesli ob izdelavi trase, pod potekom pa je izrisana tabela s podatki o vzdolžnem poteku trase, ki si jih bomo podrobneje ogledali kasneje.

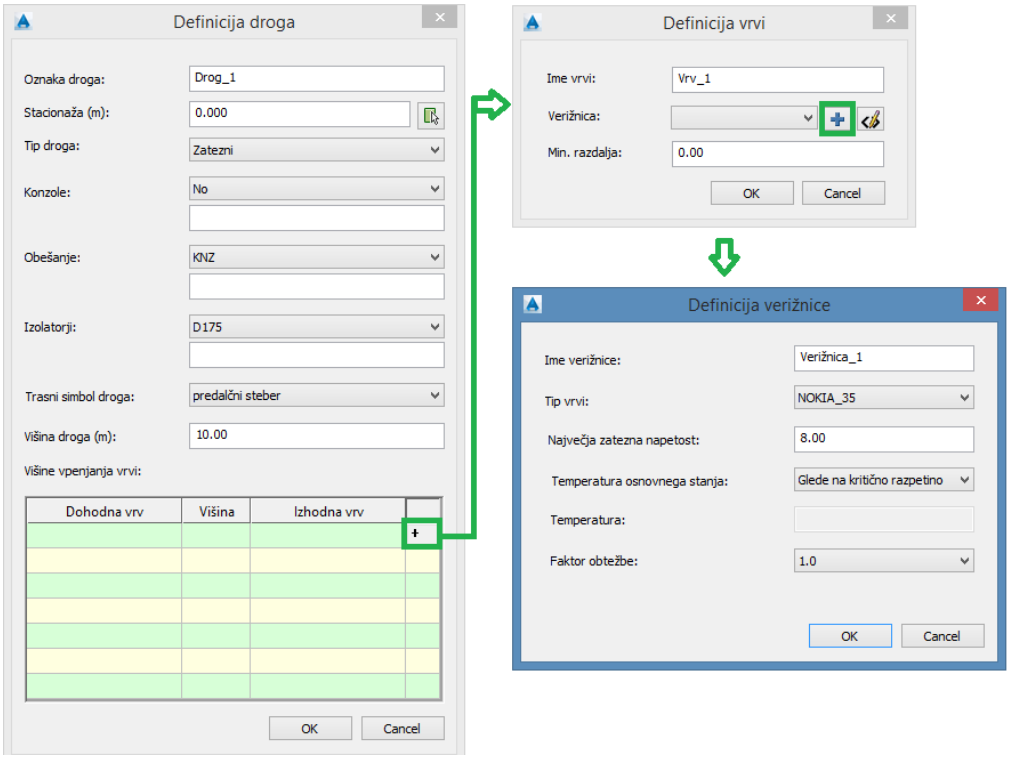


Slika 4.15: Samodejno zgeneriran vzdolžni potek vnešene trase

4.4.4 Definiranje vrvi in parametrov verižnice

Naslednji korak pri izdelavi daljnovoda je definiranje vrvi, ki so obešene preko drogov. Vsaka vrv je napeta med dvema zateznima drogovoma (ki sta lahko tudi kotna), se pravi, da vsaka vrv poteka preko enega celega zateznega polja, na vsakem zateznem polju pa je seveda lahko definiranih več vrvi. Vrv vedno definiramo na drogu, ki začenja zatezno polje, vrv pa je potem samodejno speljana preko vseh vmesnih nosilnih drogov do naslednjega zateznega droga. Osnovne lastnosti vsake vrvi so določene z definicijo verižnice. Več vrvi lahko ima in ponavadi tudi ima isto definicijo verižnice, zato si program vsako novo definicijo verižnice zapomni, tako da jo lahko uporabimo pri več vrveh. Vrv lahko vnesemo na dva načina, in sicer tako, da z orodnega traku izberemo ukaz za urejanje drogov ali pa ukaz za definiranje vrvi. V obeh primerih program od nas zahteva izbiro droga, na katerem želimo, da se vrv začne. Izberemo npr. prvi drog in prikaže se nam pogovorno okno za urejanje lastnosti droga (slika 4.16). Vrvi določamo v tabeli v spodnjem delu okna. S klikom na gumb ob desnem stolpcu razpredelnice začnemo postopek dodajanja nove vrvi. Pojavi se pogovorno okno za definicijo vrv,i v katerem lahko določimo naslednje parametre:

- ***Ime vrvi*** - vpišemo željeno ime vrvi ali pa pustimo ime, ki ga je predlagal program



Slika 4.16: Postopek dodajanja vrvi

- ***Verižnica*** - iz seznama obstoječih definicij parametrov verižnic izberemo željeno definicijo. Na tem mestu lahko uredimo parametre, katere od obstoječih definicij ali pa dodamo novo definicijo.

- ***Min. razdalja*** - vnesemo minimalno razdaljo vrvi nad terenom. Program nam bo v prikazu vzdolžnega poteka na tej oddaljenosti od vrvi izrisal pomožno vzporedno linijo, s katero bomo lahko vizualno preverili, če poteka vrv preblizu terena ali katerega drugega objekta.

V primeru da smo se odločili urejati obstoječo ali dodati novo definicijo parametrov verižnice, se odpre še pogovorno okno za definiranje parametrov verižnice, v katerem določimo sledeče parametre:

- ***Ime verižnice*** - vpišemo ime definicije parametrov verižnice ali pustimo ime, ki ga je izbral program

- ***Tip vrvi*** - iz seznama izberemo enega izmed preddifiniranih tipov vrvi. Tipi vrvi, ki so v seznamu, so definirani v ločeni datoteki xml ter vsebujejo poleg imena še podatke o površini prereza vrvi, površini prereza jedra vrvi, premer vrvi, specifično težo vrvi, težo vrvi na kilometer ter faktorja E in alpha.

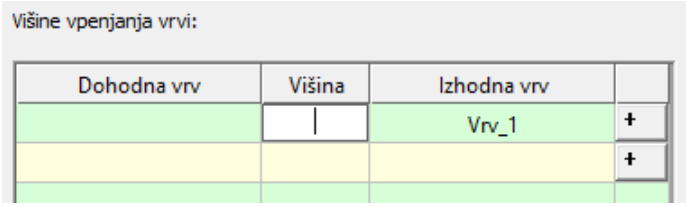
- ***Največja zatezna napetost*** - Vpišemo največjo dovoljeno zatezno napetost, ki se bo uporabljala pri izračunih.

- ***Temperatura osnovnega stanja*** - izberemo, ali se temperatura osnovnega stanja v izračunih določi glede na kritično razpetino, in sicer na $-5°C$, če je idealna razpetina večja od kritične in na $-20°C$, če temu ni tako, ali pa izberemo možnost poljubnega določanja osnovne temperature in jo določimo v spodnjem okencu.

- ***Temperatura*** (glej prejšnjo alinejo)

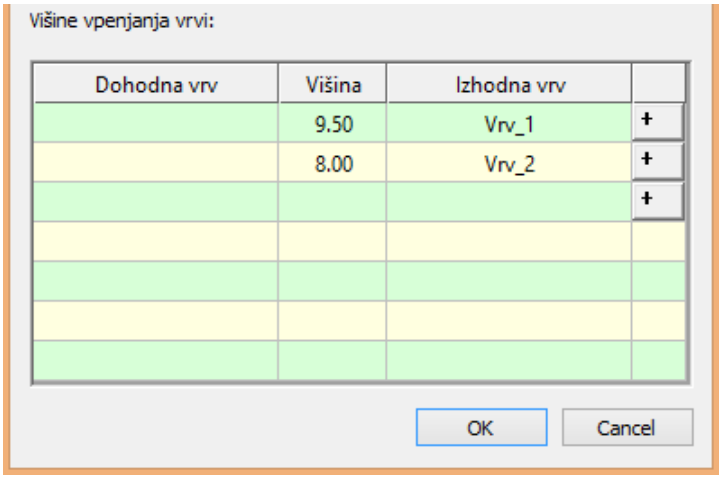
- ***Faktor obtežbe*** - izberemo enega izmed faktorjev obtežbe za izračun dodatnega zimskega bremena

Po zaključenem postopku kreiranja vrvi se ime vrvi pojavi v razpledelnici v desnem stolpcu, kjer so izpisane izhodne vrvi. Definirati je potrebno še višino obesišča vrvi, kar storimo z vpisom v polje v srednjem stolpcu razpredelnice, kot je prikazano na sliki 4.17. S tem smo definirali vrv, ki se začne



Slika 4.17: Definicija višine obesišča.

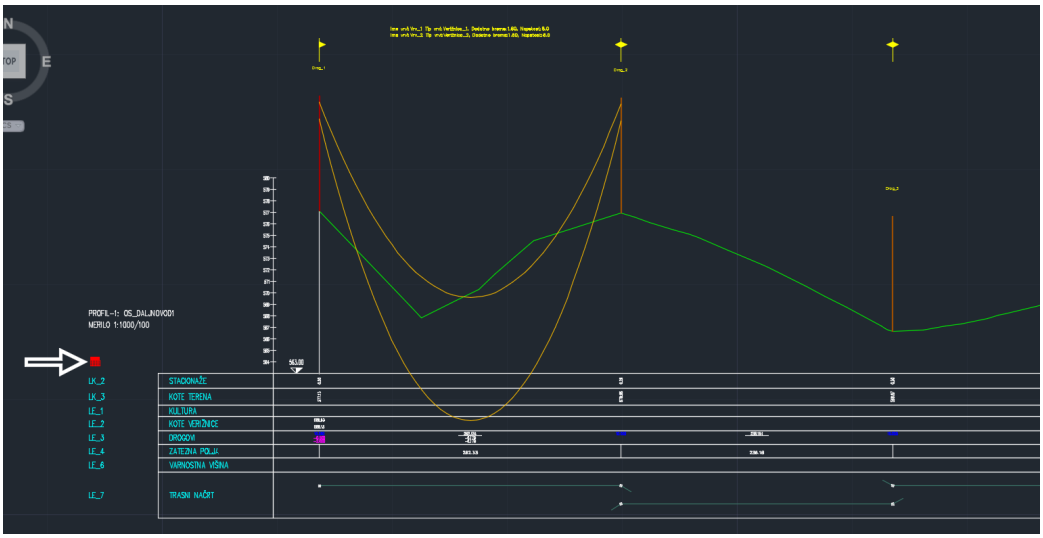
na izbranem drogu in poteka do naslednjega zateznega droga. Na naslednjem zateznem drogu se pojavi na seznamu dohodnih vrvi na prvem izmed prostih obesišč, če pa prostega obesišča ni, se kreira novo obesišče s privzeto višino, ki je enaka tisti iz začetka vrvi. Po enakem postopku lahko dodamo poljubno število vrvi na drog. V našem primeru bomo na prvem drogu definirali dve vrvi. Prva bo imela obesišče na višini 9.5 m, druga pa na višini 8 m (slika 4.18) S potrditvijo na gumb OK se izrisani vzdolžni potek trase samodejno



Slika 4.18: Dve vrvi, z začetkom na prvem drogu

posodobi in med prvima dvema drogovoma se izrišeta dve vrvi v obliki povesnih verižnic, izračunanih na podlagi vnesenih parametrov. Rezultat je viden

na sliki 4.19. Kot lahko opazimo, sta obe vrvi povešeni pod linijo terena, kar seveda ni v redu, saj morajo vrvi potekati v celoti nad določeno minimalno višino nad terenom. Ta problem bomo rešili s postavitvijo dodatnih nosilnih drogov vzdolž poteka vrvi in na ta način zmanjšali povese vrvi v sprejemljive meje.



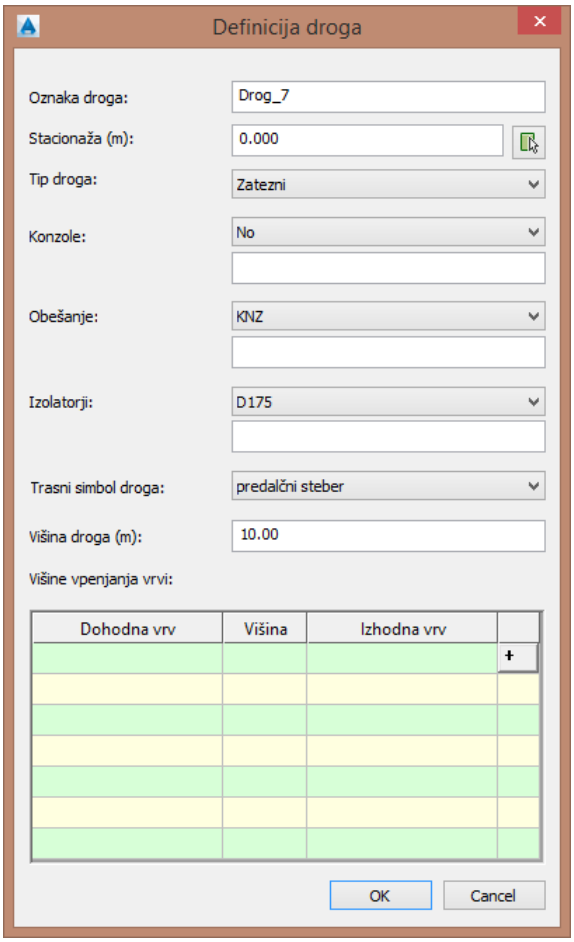
Slika 4.19: Prikaz poteka definiranih vrvi v vzdolžnem pogledu pred dodanimi nosilnimi drogovi

4.4.5 Definiranje drogov

Postopek postavitve droga na traso začnemo z izbiro ukaza za vnos drogov iz ukaznega traku. Prikaže se pogovorno okno za definicijo droga, katerega spodnji del z razpredelnico smo že spoznali pri dodajanju vrvi na drog (slika 4.20). V pogovornem oknu lahko določimo sledeče parametre:

- ***Oznaka droga*** - V polju je predlagana oznaka droga, ki jo lahko zamenjamo s poljubnim imenom.

- ***Stacionaža*** - V polje vpišemo stacionažo, na katero bi radi postavili



Slika 4.20: Pogovorno okno za definicijo droga

drog, oziroma s klikom na gumb za izbiro stacionaže preklopimo na interaktivno izbiro stacionaže (glej paragraf 4.4.5).

- ***Tip droga*** - Iz padajočega seznama izberemo ali je drog zatezni ali nosilni.

- ***Konzole, Obešanja, Izolatorji*** - V ta polja vpišemo tipe uporabljenih konzol, obešanj in izolatorjev. Ustrezna polja lahko izpolnimo s pomočjo izbire vnaprej določenih vrednosti iz padajočih seznamov. Vrednosti v teh poljih nimajo nobenega vpliva na izračune, temveč služijo le za ustrezno označevanje v risbi in projektni dokumentaciji.

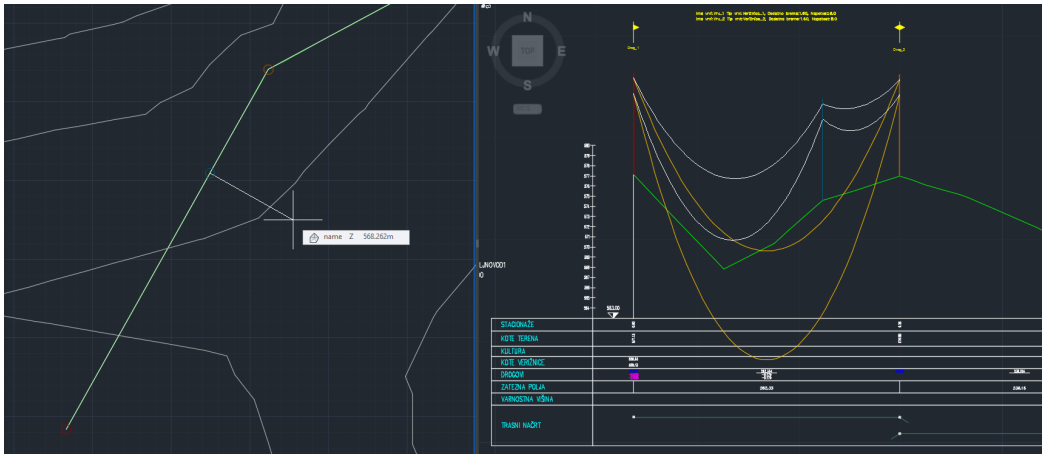
- ***Trasni simbol droga*** - Iz padajočega menija izberemo za katero od poznanih oblik droga gre. Glede na izbrano vrednost se za označevanje droga uporabi ustrezni grafični simbol v trasnem načrtu, ki se prikazuje v tabeli vzdolžnega poteka.

- ***Višina droga*** - V polje vpišemo željeno višino droga v metrih. Privzeta vrednost je 10 m.

Interaktivna izbira stacionaže droga

V pogovornem oknu za definicijo droga lahko za določitev stacionaže droga izberemo možnost interaktivnega izbiranja stacionaže. Pri tem načinu s premikanjem miške vzdolž trase izbiramo najbolj primerno mesto za postavitev droga, pri čemer v prikazu vzdolžnega poteka sproti vidimo, kakšen je vpliv položaja droga na poteke vrvi v obliki povesnih verižnic. Stacionažo lahko interaktivno izbiramo tako v vzdolžnem pogledu kot v tlorisnem pogledu. Program avtomatsko izbere pogled, ki je bližje miškinemu kazalcu v trenutku izbire. Za še večjo interaktivnost in dinamiko pri določanju stacionaže si lahko razdelimo risalno ravnino na dve vidni polji. V prvem vidnem polju imejmo tlorisni potek trase, v drugem pa vzdolžni potek trase. Na ta način bomo pri premikanju droga v enem pogledu hkrati lahko spremljali še, kakšne učinke ima premik v drugem pogledu. Na sliki 4.21 lahko vidimo,

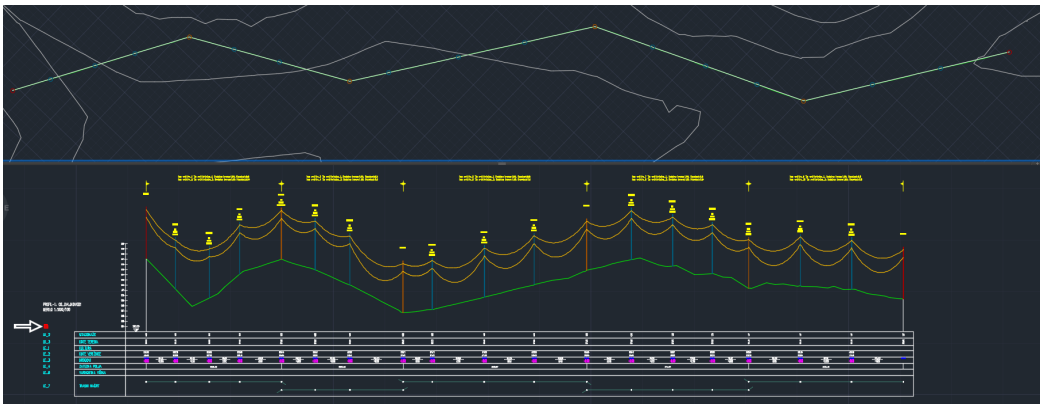
kako s premikanjem miškinega kazalca v levem vidnem polju spreminjamo položaj nosilnega droga (modri krogec) v tlorisnem pogledu, hkrati pa lahko v desnem vidnem polju spremljamo premikanje tega istega nosilnega droga (modri drog) ter predviden potek vrvi (v beli barvi) v vzdolžnem poteku. Seveda bi lahko z miško premikali tudi drog v vzdolžnem pogledu in spremljali položaj droga v tlorisnem pogledu.



Slika 4.21: Interaktivno določanje stacionaže droga

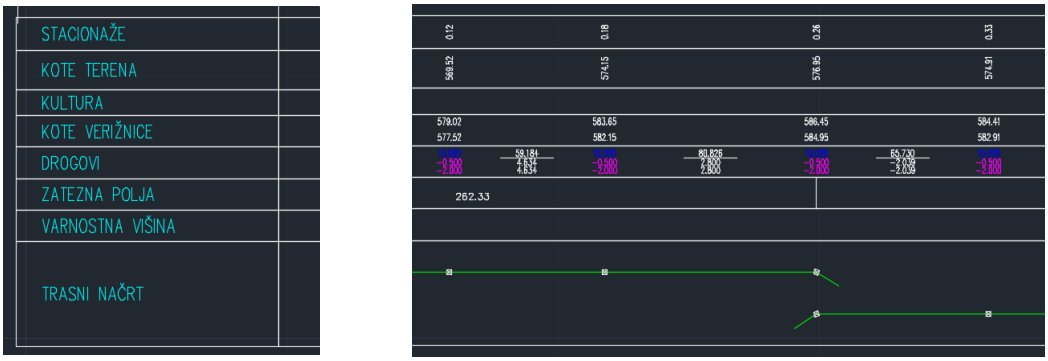
4.4.6 Tabela vzdolžnega profila

Pod vzdolžnim potekom trase daljnovoda je prikazana dinamična tabela s podatki, ki jih projektant potrebuje pri načrtovanju oziroma so zahtevani v projektni dokumentaciji projekta. Večina podatkov, ki se izpisujejo v tabelo, se dinamično spreminja ob vsaki spremembi na trasi daljnovoda, ne glede na to, če je bila sprememba narejena v tlorisnem poteku ali v vzdolžnem poteku. Na ta način je uporabniku programa zagotovljeno, da so podatki v tabeli ažurni v vsakem koraku načrtovanja, kar občutno pripomore k hitrosti in učinkovitosti uporabe programa. Za namen predstavitve tabele vzdolžnega profila smo najprej postavili še preostale drogove in vrvi po celotni trasi daljnovoda po zgoraj opisanih postopkih in kot rezultat dobili tlorisni in vzdolžni potek, ki sta prikazana na sliki 4.22.



Slika 4.22: Izgled dokončanega daljnovoda v tlorisnem in vzdolžnem pogledu

Tabela vzdolžnega profila je sestavljena iz več rubrik, ki prikazujejo določeno informacijo o daljnovodu vzdolž trase. Imena rubrik se nahajajo na levi strani tabele, vsebina pa na desni pod vzdolžnim potekom, kot prikazujeta sliki 4.23a in 4.23b. V nadaljevanju si podrobneje poglejmo, katere rubrike nastopajo v tabeli in katere informacije se prikazujejo v posameznih rubrikah.



(a) Rubrike

(b) Izsek iz vzdolžne tabele

Slika 4.23: Tabela vzdolžnega profila

- V rubriki *STACIONAŽE* je pod vsakim drogom izpisana njegova stacionaža.
- V rubriki *KOTE TERENA* je pod vsakim drogom izpisana višina terena

v točki, kjer je postavljen drog.

- Rubrika *KULTURA* je prazna in je zaenkrat namenjena temu, da lahko uporabnik vanjo vpiše katastrsko kulturo zemljišča za posamezne odseke trase
- V rubriki *KOTE VERIŽNICE* so pod vsakim drogom izpisane višine obesišč posamezne vrvi na drogu.
- V rubriki *DROGOVI* je pod vsakim drogom izpisana višina droga ter relativna višinska razlika obesišča vsake vrvi od vrha droga
- V rubriki *ZATEZNA POLJA* so izpisane dolžine zateznih polj daljnovoda. To pomeni da je med vsakima dvema zaporednima zateznima ali kotnima drogovoma izpisana oddaljenost med njima.
- V rubriki *VARNOSTNA VIŠINA* se za izbrane točke vzdolž trase izpisuje višinska razlika med terenom v tej točki in med najnižjim potekom vrvi v tej točki. Točke, v katerih želimo imeti tabelirano varnostno višino, izbiramo s pomočjo ukaza za tabeliranje varnostnih višin, ki se nahaja na orodnem traku (slika: 4.6). Po izbiri ukaza se v ukazni vrstici izpišejo možnosti, ki so na voljo za določanje točk tabeliranja, kot je prikazano na sliki 4.24.

Slika 4.24: Tabeliranje varnostne višine.

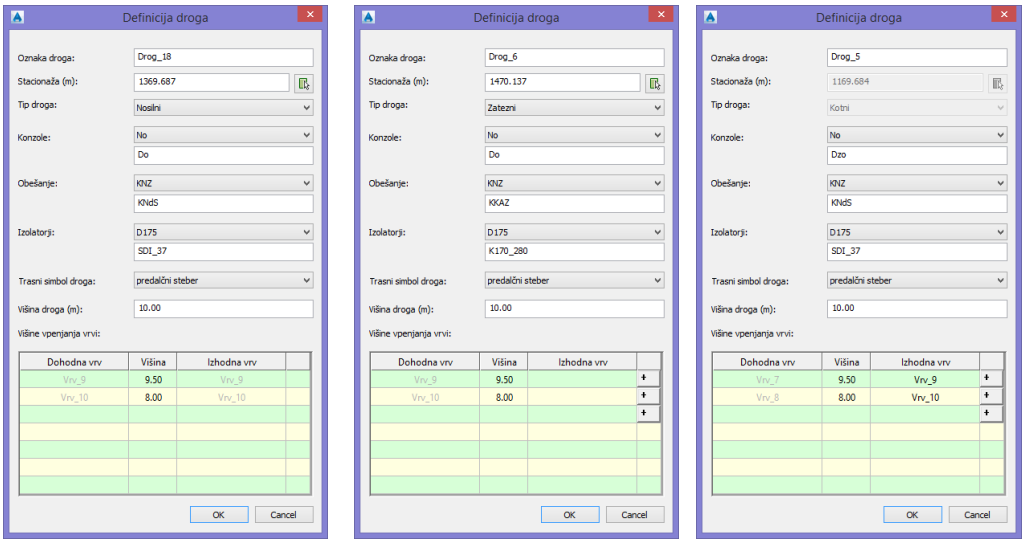
  - ***Točka tabeliranja*** je privzeta možnost, ki uporabniku omogoča, da točke izbira poljubno s klikanjem vzdolž trase v vzdolžnem prikazu.
  - ***Stacionaža*** je možnost, ki omogoča določanje točke tabeliranja z vpisom poljubne stacionaže.
  - ***Razdalja*** je možnost, ki omogoča določanje točk tabeliranja z vpisom razdalje koraka s katerim se dodajo točke vzdolž trase.

  - ***Profili*** je možnost, ki omogoča določanje točk tabeliranja na vseh mestih, kjer je postavlje drog.
  - ***Teren*** je možnost določanja točk v vseh prelomih terenske črte v vzdolžnem poteku.
  - ***Briši*** je možnost, ki omogoča brisanje seznama točk tabeliranja.

- V rubriki *TRASNI NAČRT* je izrisan načrt trase.Trasni načrt je sestavljen iz večih odsekov. Vsak odsek predstavlja del trase med dvema horizontalnima lomoma trase daljnovoda. Odsek trase je prikazan z ravno linijo, ki ima na vsakem koncu še shematični podaljšek pod realnim kotom loma trase do prejšnjega oziroma naslednjega odseka trase. Na mestih, kjer stoji drog, je izrisan trasni simbol droga glede na obliko droga, ki smo jo določili ob definiciji droga (slika: 4.20)

4.4.7 Dinamično urejanje daljnovoda

Ena od glavnih prednosti programa, ki smo jo želeli zagotoviti, je dinamično urejanje trase daljnovoda. Nekaj te dinamike smo predstavili že v poglavju 4.4.5 pri vnašanju drogov. Urejanje drogov namreč opravljamo prek enakega uporabniškega vmesnika kot vnašanje novih drogov.
Iz ukaznega traku izberemo ukaz "*Urejanje drogov*", ki od nas zahteva izbiro droga. Drog lahko izberemo tako v tlorisnem kot v vzdolžnem pogledu, nakar se pojavi enako pogovorno okno kot pri vnosu drogov (slika 4.25), le da ima že vnešene vse podatke o izbranem drogu vključno s podatki o vrveh. V pogovornem oknu lahko ustrezno spremenimo parametre izbranega droga, vendar pa obstajajo določene omejitve glede na tip izbranega droga. Če je izbrani drog nosilni, mu lahko spreminjamo stacionažo, ne moremo pa mu spreminjati podatkov o vrveh. Vrvi, ki so obešene na nosilni drog, so v razpredelnici izpisane, ne moremo pa jih urejati, saj so definirane na prvem predhodnem zateznem ali kotnem drogu.



(a) Nosilni drog (b) Zatezni drog (c) Kotni drog

Slika 4.25: Pogovorna okna za urejanje treh tipov drogov

Tip droga lahko spremenimo le iz zateznega v nosilni in obratno. Pri spremembi tipa droga stopijo v veljavo prej opisane omejitve glede urejanja določenih parametrov, poleg tega pa se spremeni tudi sam potek vrvi na območju tega droga.

Pri spremembi nosilnega droga v zatezni se vse vrvi, ki so prej potekale preko tega droga, zdaj zaključijo na njemu. Vrvi, ki jih želimo od tega droga naprej, moramo definirati na tem drogu.

Pri spremembi zateznega droga v nosilni se vse vrvi, definirane na tem drogu, izbrišejo, vse dohodne vrvi, ki so se prej zaključile na tem drogu, pa se po novem nadaljujejo preko njega do naslednjega zateznega oziroma kotnega droga.

Kotnemu drogu ne moremo spremeniti tipa, saj mora ostati kotni, ker je na lomu trase. Ravno tako tudi ne moremo katerikoli drug tip droga spremeniti v kotnega.

Vsaka sprememba parametrov droga oziroma vrvi se samodejno odrazi v ustrezni posodobitvi prikaza vzdolžnega poteka daljnovoda, tabele vzdolžnega poteka ter prikaza tlorisnega poteka daljnovoda. Največ dinamike je pri premikanju drogov z interaktivno izbiro stacionaže, saj med premikanjem droga hkrati opazujemo spreminjanje njegove pozicije v tlorisnem pogledu, kakor tudi v vzdolžnem pogledu. Tako lahko uporabnik programa pazi, da ne bo droga postavil na neustrezno parcelo, cesto ali kak drug objekt, hkrati pa spremlja, kaj se dogaja s povesi vrvi v vzdolžnem poteku in pazi, da se ohranja minimalna varnostna višina pod vrvmi.

Urejanje tlorisnega poteka trase je ravno tako problematika, kjer pride do izraza možnost dinamičnega urejanja. Pri spremembi poteka trase daljnovoda se spremeni več parametrov daljnovoda. Sama trasa se s spremembo lahko podaljša ali skrajša, lahko se spremeni število lomov trase in s tem število kotnih drogov, poleg tega se spremeni višinski potek terenske linije v vzdolžnem poteku, saj je to projekcija trasne polilinije na terensko površino. Iz tega je razvidno, da pomeni vsaka sprememba poteka trase zelo veliko spremembo vzdolžnega prereza. Zaradi tega je bilo ključno, da v program vgradimo funkcionalnost, ki omogoča čimbolj enostavno in hitro urejanje poteka trase daljnovoda ter pri tem avtomatsko ažurira vse podatke in prikaze. Uporabniku programa smo omogočili, da urejanje poteka trase izvede kar z urejanjem polilinije trase z uporabo AutoCAD-ovih ukazov in načinov za urejanje polilinije. Program ima na polilinijo trase vezan reaktor, ki zazna vsako spremembo, ki se zgodi na poliliniji in ob tem izvede preračun celotnega daljnovoda vključno s projekcijo novega poteka trase na terensko površino in ažuriranjem tabele vzdolžnega prereza. Pri tovrstni spremembi se ponovno izračunajo stacionaže kotnih drogov ter po potrebi dodajo novi ali izbrišejo obstoječi. Ostalim drogovom se stacionaža ne spremeni. Drogovi, katerih stacionaža je večja, kot je dolžina nove trase, se izbrišejo.

4.4.8 Generiranje poročila

Po končanem načrtovanju daljnovoda nam program omogoča generiranje poročila v tekstovni obliki. V tem poročilu so zbrane montažne tabele ter nekateri drugi pomembnejši parametri daljnovoda. Montažne tabele vsebujejo podatke o povesih vrvi in nateznih silah pri različnih temperaturah in se uporabljajo pri napenjanju vrvi na daljnovod. V poročilu so za vsako vrv daljnovoda navedeni osnovni podatki, ki so bili uporabljeni v izračunu, ter nato izračuni in montažne tabele za vsako razpetino, kjer ta vrv nastopa. V primeru 4.8 lahko vidimo izsek iz poročila za daljnovod na sliki 4.8 za prvo razpetino vrvi z imenom Vrv 1.

```
Ime vrvi                    : Vrv_1
Tip vrvi                    : AlFe_70/12
Idealna razpetina           :   67.655 m
Kritična razpetina          :   43.358 m
Osnovna temperatura         :  -5.0 st.C
Kritična temperatura        :  37.7 st.C
Največji poves pri temp.    :  40.0 st.C
Osnovna napetost            :8.00 daN/mm2
Faktor dodatnega bremena    :       1.6
Skupna dolžina verižnice    :     262.9

Razpetina                   :Drog_1 - Drog_8
Poves pri max. napetosti    :0.785 m
Nateg                       :8.102 daN/mm2
Natezna sila                :658.676 daN
Največji poves              :0.803 m
```

Temp	Poves	Dolžina	Napetost	Nateg	Rez.napetost	Rez.nateg
St.C	m	m	daN/mm2	daN		
-20	0.268	56.546	5.253	427.109	5.293	430.345
-15	0.300	56.547	4.694	381.594	4.731	384.592
-10	0.336	56.548	4.181	339.932	4.215	342.714
-5	0.378	56.550	3.724	302.771	3.756	305.363
0	0.423	56.551	3.327	270.475	3.357	272.905
5	0.471	56.553	2.989	243.016	3.017	245.309
10	0.520	56.556	2.706	220.008	2.733	222.191
15	0.569	56.558	2.471	200.856	2.496	202.949
20	0.618	56.561	2.274	184.906	2.299	186.926
25	0.667	56.564	2.110	171.552	2.134	173.513
30	0.713	56.567	1.971	160.280	1.995	162.194
35	0.759	56.570	1.853	150.676	1.876	152.550
40	0.803	56.573	1.752	142.412	1.774	144.254
-5	0.785	56.880	8.000	650.400	8.102	658.676

Primer 4.8: Izsek iz poročila prikazuje montažne tabele za prvo razpetino vrvi Vrv 1

Poglavje 5

Sklepne ugotovitve

V okviru diplomske naloge smo razvili aplikacijo Electra za načrtovanje nadzemnih elektroenergetskih vodov, ki smo jo integrirali v obstoječo programsko opremo CGS Infrastructure Design Suite kot dodatni modul.Uspešno smo implementirali vse točke zahtevane funkcionalnosti iz **??**. Pri razvoju smo poleg zagotavljanja natančnosti in pravilnosti izračunov največ poudarka namenili razvoju funkcionalnosti, ki omogoča intuitivno in dinamično interakcijo uporabnika s programom. Poskusili smo doseči, da bi uporabnik večji del interakcije s programom in s podatki opravil preko same risbe DWG in v manjši meri preko drugih oblik uporabniškega vmesnika, saj je takšen način dela bolj naraven ter v duhu programov CAD, ki so osredotočeni na vizualno načrtovanje. Druga stvar, ki smo ji posvetili precej pozornosti, pa je ažurna skladnost vzdolžnega in tlorisnega pogleda načrta nadzemnega elektroenergetskega voda. Da nam je oboje dobro uspelo, gre v veliki meri zasluga tudi zelo dobremu programskemu vmesniku ObjectARX, ki smo ga s pridom izkoristili za dostop do funkcionalnosti, ki so na voljo v samem Autocad-u za delo s podatkovnimi in vizualnimi elementi risbe DWG.
Program Electra je že v uporabi v enem od slovenskih podjetij za distribucijo električne energije, s svojo funkcionalnostjo pa cilja na vse, ki se ukvarjajo z načrtovanjem ali vzdrževanjem nadzemnih elektroenergetskih vodov in so uporabniki platforme Autocad ali Bricscad.

Možne nadgradnje in izboljšave programa Electra so številne in bodo v večeni implementirane v prihodnjih različicah programa. Naštejmo nekaj najpomembnejših:

- podpora večim evropskim in svetovnim standardom načrtovanja nadzemnih elektroenergetskih vodov
- vpliv sile vetra na vodnike in ostale elemente nadzemnega elektroenergetskega voda
- možnost prečnih prerezov trase daljnovoda
- možnost 3D pogleda nadzemnih elektroenergetskih vodov
- podpora projektiranju podzemnih elektroenergetskih vodov

Literatura

[1] F. Kiessling, P. Nefzger, J.F.Nolasco, U. Kaintzyk *Overhead Power Lines: Planning Design Construction*, New York: Springer, 2003, pogl. 14.

[2] I. Papič, P. Žunko, *Elektroenergetska tehnika I*, Ljubljana: Fakulteta za elektrotehniko, 2005, pogl. 3.

[3] M. Plaper *Elektroenergetska Omrežja III. del*, Ljubljana: Fakulteta za elektrotehniko, 1977

[4] (2015) ObjectARX for AutoCAD 2013: Developer's Guide. Dostopno na:
http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=785550

[5] (2015) PLS-CADD (Power Line Systems - Computer Aided Design and Drafting). Dostopno na:
http://www.powline.com/products/pls_cadd.html

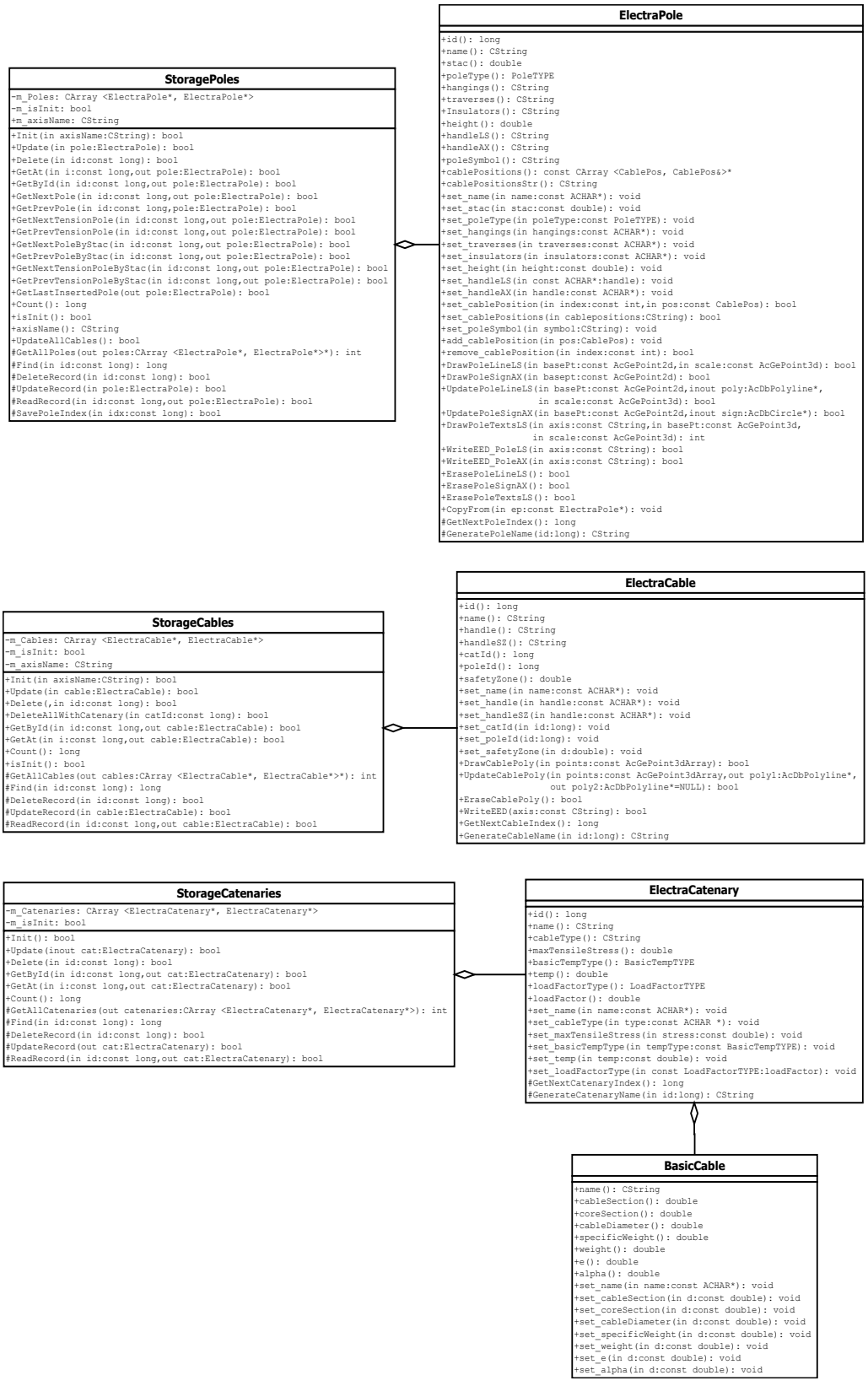
[6] (2015) CATAN, Software for Overhead Power Line Design. Dostopno na:
http://www.catanlines.com.au/CATANSoftware/CATAN_Software.htm

[7] (2015) Livewire 3 - Software for the Mechanical Design of Pole Type Power Lines. Dostopno na:
http://www.power-technology.com/contractors/front/rob-wilks/

[8] (2015) Poles ‘n’ Wires, QUICK AND SIMPLE POWER LINE DESIGN SOFTWARE. Dostopno na:
http://ipowermation.com/media/pnwbrochure.pdf

Dodatek A

Razredni diagram podatkovnega sloja programa Electra



Slika A.1: Razredi podatkovnega sloja

Dodatek B

Uporaba funkcionalnosti AcEdJig za urejanje položaja drogov

```
AcEdJig::DragStatus ElectraLSJig::doIt() {
  //kreiranje vseh entitet, ki jih vključimo v Jig
  ...
  //inicializacija electra podatkovne baze
  ...
  //izračunaj začetne vrednosti za potek vrvi
  ...
  //nastavi barve entitet, sloje ter tipe črt
  ...
  //poženi Jig
  do {
    stat=drag();
    if ((stat==AcEdJig::kNull)) stat=AcEdJig::kCancel;
  }while((stat!=AcEdJig::kNormal) && (stat!=AcEdJig::kCancel));
  //izbriši entitete vključene v Jig
  ...
}

AcEdJig::DragStatus ElectraLSJig::sampler() {
  //pridobimo novo točko položaja miškinega kazalca
  stat=acquirePoint(m_ptIn);
  //preverimo če je sprememba dovolj velika, da je potrebno posodobiti izris
  if ((fabs(m_pt0.x-m_ptIn.x) > MIN_NUM) || (fabs(m_pt0.y-m_ptIn.y) > MIN_NUM)) {
    m_pt0=m_ptIn;
  }
  else if(stat==AcEdJig::kNormal) {
    stat=AcEdJig::kNoChange;
  }
}

Adesk::Boolean ElectraLSJig::update() {
  //izračunamo nov položaj droga v vzdolžnem profilu glede na novo točko m_ptIn
```

```
  ...
  //posodobi entiteto, ki predstavlja linijo droga, glede na nov položaj
  ...
  //s pomočjo funkcij našega razreda CableCalculation na novo izračunamo točke obešanja
     vrvi na drogove
  //ter izračunamo nove verižnice, po katerih potekajo obešene vrvi (m_cc je objekt tipa
     CableCalculation)
  ...
  m_cc->CalculateHangingPoints(&cableData->m_cab, m_scale, m_stac0, cableData->
     m_hangingPoints, m_ep->id());
  m_cc->Calc(&m_cablesData.ElementAt(i)->m_cat, &m_cablesData.ElementAt(i)->m_bc,
     hangingPoints, points, m_scale));
  ...
  //glede na izračunane verižnice, posodobimo polilinije, ki predstavljajo vrvi
  ...
  //posodobljene entitete dodamo v seznam entitet, ki jih je potrebno posodobiti
  entityList.append(m_pLineSitu); entityList.append(m_pLine); entityList.append(
     m_pCircle);
  m_EntList.setEntityList(entityList);
}

AcDbEntity* ElectraLSJig::entity() const {
  return(const_cast<CEntityList*>(&m_EntList));
}

Adesk::Boolean CEntityList::subWorldDraw(AcGiWorldDraw * wd) {
  //nad vsemi entitetami v  seznamu m_EntList, pokličemo wd->geometry().draw(ent);
  //zato, da se posodobljene entitete ponovno izrišejo
}
```

Primer B.1: Uporaba funkcionalnosti AcEdJig za urejanje položaja drogov